i tuoi amici commodore 16 e PLUS 4

Brian Lloyd



EDIZIONE ITALIANA



GRUPPO EDITORIALE JACKSON

i tuoi amici commodore 16 e PLUS 4

Brian Lloyd



GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

TRADUZIONE: Paola Saini
GRAFICA E IMPAGINAZIONE: Moreno Confalone
SUPERVISIONE TECNICA: Vittorio Riva
REVISIONI E PROGRAMMI: Marco Saini
COPERTINA: Silvana Corbelli
FOTOCOMPOSIZIONE: System Graphic
STAMPA: Grafika 78

INDICE

INDICE DETTAGLIATO

CAPITOLO 7
TUTTI FACCIAMO DEGLI ERRORI
Intercettazione degli errori •HELP • TRON e TROFF.

CAPITOLO 8
PROGRAMMAZIONE AVANZATA
READ, DATA e RESTORE • Uso delle stringhe • LEFT$ • RIGHT$ • MID$ • INSTR • LEN • SOUND e VOLume • Valore delle note musicali • ON ... GOTO e ON ... GOSUB • AUTO • CLEAR • ASC • VALSTR$.

CAPITOLO 9
STAMPA E GRAFICI
PRINT USING • PUDEF • Grafici • COLOR • GRAPHIC • LOCATE • DRAW • BOX • SCALE • PAINT • Grafici multicolore • SSHAPE e GSHAPE • RCLR • RGR • RDOT • Programma ARTISTA • Come funziona il programma.

CAPITOLO 10
FUNZIONI
DEF FN tasti funzione • Funzioni numeriche: ABS • DEC • EXP • LOG • SGN • SQR • USR • Funzioni trigonometriche • Altre funzioni: HEX$ • FRE • POS • SPC.

CAPITOLO 11
LINGUAGGIO MACCHINA
PEEK e POKE • Introduzione a TEDMON • Uscita da TEDMON • Riempire un'area della memoria • Ricerca di numeri e stringhe • Trasferimento di blocchi di memoria • Programmi scritti in linguaggio macchina • Esecuzione di programmi in linguaggio macchina • Disassemblare programmi in linguaggio macchina • Confronto di blocchi di memoria • Salvare programmi in linguaggio macchina • Caricare programmi in linguaggio macchina • Verificare programmi in linguaggio macchina • I registri • Il comando SYS • La funzione USR.

CAPITOLO 12
PERIFERICHE
Uso dell'unità dischi • Precauzioni • Inizializzazione dei dischetti • La DIRECTORY del dischetto • Salvare un programma • Controllo del programma • Caricamento del programma • Cambiare il nome del programma • Fare una copia in più del programma • Cancellare un programma dal dischetto • Salvare di nuovo un programma • Riordino del dischetto • Copia di un dischetto • Uso della stampante • Uso di file su nastro • Uso di file su dischetto.

INTRODUZIONE ALL'EDIZIONE ITALIANA

La guida di Brian Lloyd è un valido strumento per chi desidera incominciare a programmare in BASIC con il Commodore 16 o Plus 4, senza essere affatto uno studio didattico o un testo sul BASIC. Il lettore viene introdotto, passo dopo passo, nella programmazione attraverso esempi chiari ed esaurienti, che rappresentano un saggio sull'uso del computer. Questo sistema spinge a provare cose sempre nuove, unendo l'utile al dilettevole e facendo del proprio computer uno strumento operativo dalle possibilità quasi illimitate. Il lettore scoprirà le capacità grafiche ed artistiche del C16/Plus 4 insieme alle capacità matematiche. Può risolvere infatti vari tipi di calcoli, dalle operazioni più semplici alle funzioni più avanzate. Ma non c'è bisogno di essere un genio matematico, anzi tutto appare semplice, lineare e facile da comprendere.

Il libro è destinato al principiante, cioè a colui che è completamente "digiuno" di computer e a chi, conoscendo già il BASIC, vuole solo imparare il BASIC Commodore 3.5.

Il principiante ha a sua disposizione una spiegazione dettagliata di tutti i comandi, mentre chi è già esperto, può far riferimento a delle Appendici sintetiche ma esaurienti.

Il libro si compone di 12 capitoli e mentre i primi servono a creare una conoscenza tra il computer e il "novello" programmatore, quelli successivi approfondiscono le molteplici e più interessanti capacità dell'hardware, stimolando nel lettore il desiderio di apprendere e concretizzare l'utilizzo della "macchina" che presto diventerà una inseparabile amica.

Infine, il glossario inserito tra le Appendici, in cui sono raccolti i termini che fanno parte del linguaggio del computer e che perciò, non sono di uso corrente.

È quindi consigliabile un esame attento ed accurato del Manuale perché le parti sono fra di loro concatenate e nel loro insieme danno una completa panoramica dell'argomento.

Paola Saini

PREFAZIONE

Questo libro è stato scritto appositamente per chi è completamente digiuno di computer e questo scopo è rimasto sempre nella mente dell'autore durante lo svolgimento dell'opera. Questo fattore non impedirà però al programmatore esperto, di apprendere la versione del BASIC usata con i computer Commodore Plus/4 e C16 perché ogni comando viene spiegato in modo chiaro ed esauriente. Imparerai ad usare il computer e a fare tutto quello che desideri, perché ogni cosa è spiegata minuziosamente a partire da quella più semplice quale ad esempio il collegamento iniziale del computer, fino ad arrivare a quelle più complicate quali ad esempio le tecniche di programmazione più sofisticate con l'uso dei file e dei comandi PEEK e POKE.

Sebbene tutto sia spiegato nei particolari, ho cercato di evitare lunghe ed inutili disquisizioni sugli argomenti più semplici e comunque il libro è progettato in modo da permetterti di programmare fin dall'inizio. I comandi vengono spiegati secondo le tue necessità per non farti fare confusione prima di aver imparato ad usare bene la tastiera.

Ti consiglio di leggere attentamente il libro, senza saltare pagine solo perché sembra magari una lettura noiosa. Certo è difficile a farsi, perché è più che naturale che tu sia ansioso di andare avanti per provare cose sempre nuove. Sfortunatamente, agendo in questo modo non ottieni niente, anzi perdi tempo; perciò è meglio che tu studi il libro lentamente e ordinatamente se vuoi realmente diventare un buon programmatore. Se non capisci bene qualcosa alla prima lettura, torna indietro e riguardala.

La cosa più importante è la pratica ed infatti è il mezzo migliore e più efficace per imparare. Utilizza nella pratica tutto quello che impari per vedere come funziona (e anche perché non funziona!). Non aver paura di danneggiare il computer perché anche se lo colpissi con forza non potresti fargli realmente del danno.

Infine, prima di iniziare la lettura, devi sapere che il linguaggio principale usato dal tuo computer e, quello che questo libro si propone di insegnarti, è chiamato BASIC. BASIC è una sigla (Beginner's All Purpose Symbolic Instruction Code) e significa codice di istruzioni simboliche (MULTIUSO) per il principiante, e anche se può sembrare un po' complicato, ti rassicurerà sapere che la prima parola è principiante (Beginner). Ma non devi scoraggiarti: anche se il BASIC è un linguaggio per principianti è comunque molto efficace.

RINGRAZIAMENTI

La mia gratitudine va allo staff della Commodore Ltd. specialmente a Gail Wellington, Phil Gosling e allo staff di tecnici, senza la cui pazienza e senza il cui aiuto, questo libro non sarebbe mai stato scritto.

CAPITOLO 1

I PRIMI PASSI

Il collegamento

Il primo e più importante passo per imparare come usare il tuo Commodore Plus/4 o C16 è sapere come collegarlo. Bisogna collegarlo con una presa elettrica e con una televisione e più tardi con un registratore, e possibilmente una unità dischi (disk drive) e una stampante, di cui parleremo in seguito.

La prima cosa da collegare è l'alimentatore: inserisci la spina DIN alla fine del cavo dell'alimentatore nella presa denominata POWER (sulla destra per il C16, sulla sinistra per il PLUS/4, guardando da dietro), dopo esserti assicurato che l'interruttore ON/OFF sulla destra sia sulla posizione OFF. Inserisci poi l'alimentatore in una presa a muro adatta.

Dovrai avere anche un cavo tra la TV e il computer. Ad una estremità c'è una spina phono che va inserita nell'uscita RF (sulla sinistra del PLUS/4 e sulla parte posteriore del C16). L'altra estremità del cavo va inserita alla televisione al posto dell'antenna.

A questo punto non è necessario collegare niente altro, né il registratore, né il disk drive (unità dischi), né altre periferiche che potranno essere collegate quando avrai imparato ad usarle. Ora, ecco il grande momento! Accendi la televisione e poi il computer (si accenderà una spia rossa) e se non sei fortunato, lo schermo della tua televisione apparirà come se non fosse inserito il cavo dell'antenna e sentirai un rumore sibilante. Ciò che devi fare è sintonizzare la televisione con il computer, dopo di che apparirà il messaggio:

```
COMMODORE BASIC V3.5 60671 BYTES FREE
READY.
```

(questo è per il PLUS/4) o

```
COMMODORE BASIC V3.5 12277 BYTES FREE
READY.
```

(questo è per il C16).

Questo è il messaggio di partenza che ti dice che il tuo computer capisce la versione 3.5 del BASIC Commodore oltre a quanti bytes di memoria hai a disposizione (un byte contiene un carattere, ciò significa che utilizzerai 4 bytes di memoria per scrivere la parola CIAO, dato che questa è composta di 4 caratteri). La parola READY (pronto) indica che il computer è pronto per ricevere le tue istruzioni. Sotto la parola READY vedrai un quadrato luminoso lampeggiante, chiamato cursore. Si trova nel punto in cui verrà scritto il prossimo carattere che digiterai sulla tastiera.

Se hai un PLUS/4 apparirà anche il messaggio 3-PLUS-1 ROM inserito e quale tasto premere. Devi fare riferimento al Manuale 3-PLUS-1 d'istruzioni per l'uso di questa ottima facilitazione.

La tastiera

Ora passiamo alla tastiera. A prima vista sembra uno strumento molto complicato, ma non ti ci vorrà molto per imparare ad usarla. Chi ha dimestichezza con la macchina per scrivere, noterà che i tasti delle lettere sono sistemati seguendo la disposizione standard QWERTY (la disposizione viene chiamata QWERTY perché le prime sei lettere sulla tastiera sono la Q, W, E, R, T, Y). Comunque, i tasti del tuo computer hanno molti più simboli rispetto a quelli della macchina per scrivere, oltre a parecchi tasti speciali.

Di solito quando scrivi con una macchina per scrivere, appaiono sul foglio lettere minuscole, mentre se scrivi con il computer le lettere appariranno maiuscole sullo schermo della televisione. Se digiti:

```
CIAO A TUTTI
```

vedrai apparire le lettere maiuscole con il cursore che si muove lungo la linea da sinistra a destra mentre digiti.

Caratteri minuscoli

Il tuo computer può comunque scrivere con caratteri minuscoli. Se tieni premuto uno dei tasti SHIFT e premi contemporaneamente il tasto Commodore (quello in basso a sinistra con il simbolo della Commodore) vedrai cambiare tutte le lettere sullo schermo e diventare minuscole. Se ora digiti:

```
Ciao a tutti
```

vedrai le lettere scritte con caratteri minuscoli.

Quando vuoi scrivere con la maiuscola su una macchina per scrivere devi tenere premuto il tasto della maiuscola mentre digiti le lettere che desideri. Lo stesso avviene con il computer quando sei nel modo minuscolo di scrittura. Per esempio per digitare:

```
Ciao Luigi
```

devi prima assicurarti di essere nel modo di scrittura minuscolo e poi tenere premuto il tasto SHIFT mentre premi il tasto C, poi lascia lo SHIFT e digita 'iao', premi la barra spaziatrice poi tieni premuto di nuovo il tasto SHIFT mentre premi il tasto L e poi digita 'uigi'.

Quando vuoi dare al computer dei comandi noterai che possono essere inseriti in maiuscola o minuscola. Se, comunque sei nel modo di scrittura minuscolo e usi il tasto SHIFT per far diventare una lettera maiuscola e poi digiti un comando, esso viene ignorato.

Simboli maiuscoli

Il tasto SHIFT viene usato anche per ottenere i caratteri sopra i numeri. Per es. per ottenere il simbolo del dollaro devi tenere premuto il tasto SHIFT e poi premere il tasto 4. Lo stesso vale per la virgola, il punto, la sbarra, i due punti e il punto e virgola, ognuno dei quali ha due simboli sulla parte frontale in alto. Per ottenere uno qualsiasi dei caratteri in alto devi premere il tasto SHIFT e il tasto con il simbolo che desideri.

Pulizia dello schermo

D'ora in poi lo schermo sarà probabilmente quasi tutto occupato. Fortunatamente per noi, è molto facile pulire lo schermo da tutto quello che contiene. Se premi il tasto CLEAR/HOME il cursore salterà all'angolo in alto a sinistra dello schermo. Se premi i tasti SHIFT e CLEAR/HOME lo schermo verrà pulito e il cursore apparirà nell'angolo in alto a sinistra, pronto per scrivere di nuovo.

I tasti controllo cursore

Probabilmente avrai notato i tasti con quattro frecce in alto a destra. Sono chiamati tasti controllo cursore e premendone uno si fa muovere il cursore nella direzione indicata dalla freccia. Questi tasti ti permettono di scegliere in quale punto vuoi che appaia il carattere che stai digitando.

Il tasto INST/DEL

Nell'angolo in alto a destra sulla tastiera c'è il tasto INST/DEL. Esso serve per cancellare caratteri dallo schermo e inserire spazio. Per es. assicurati che il computer sia nel modo di scrittura maiuscola (ricorda: il tasto SHIFT e Commodore ti permette di passare dalla maiuscola alla minuscola e viceversa) e digita quanto segue:

```
LUIGGI CAMMINA LA STRDA
```

La prima cosa da fare ora è correggere STRDA. Per far ciò muovi il cursore sulla lettera D (usando i tasti controllo cursore) e digita ADA. Le nuove lettere rimpiazzeranno le vecchie e sul video rimarrà:

```
LUIGGI CAMMINA LA STRADA
```

Ora bisogna togliere la seconda G di LUIGGI. Spostati con il cursore sulla seconda G e premi il tasto INST/DEL. Vedrai che la G in più scomparirà e le lettere seguenti si sposteranno a sinistra di un posto, così:

```
LUIGI CAMMINA LA STRADA
```

La frase però non ha senso; bisogna inserire la parola LUNGO tra CAMMINA e LA; per far questo spostati con il cursore sullo spazio tra CAMMINA e LA e tenendo premuto il tasto SHIFT, premi 6 volte il tasto INST/DEL e lascia il tasto SHIFT. Ora puoi digitare LUNGO in modo da ottenere:

```
LUIGI CAMMINA LUNGO LA STRADA
```

Shift Lock

Vicino al tasto SHIFT di sinistra c'è il tasto SHIFT LOCK. Questo tasto fa quello che il suo stesso nome suggerisce, cioè blocca lo SHIFT in modo che tu non debba tenere premuto continuamente il tasto SHIFT per scrivere caratteri maiuscoli. Se premi una volta sola il tasto SHIFT LOCK vedrai che si blocca, circa a metà altezza ed è quindi inserito. Premendolo una seconda volta ritorni alla situazione normale.

Control

Un altro tasto con una funzione speciale è il tasto CONTROL. Il PLUS/4 ha due tasti CONTROL, ognuno ad una estremità della seconda fila di tasti. Il C16

ha solo un tasto CONTROL, ma non è uno svantaggio. Se dai uno sguardo ai tasti numerici da 1 a 8 vedrai che ognuno ha due colori differenti sulla parte frontale. Premendo il tasto CONTROL e uno di questi tasti puoi cambiare il colore del cursore (e il colore di un qualsiasi carattere che digiterai) con quello sulla parte superiore del tasto.

Il tasto Commodore

Il tasto Commodore può essere usato per selezionare i colori indicati sulla parte inferiore del tasto numerico. Per es. se premi il tasto Commodore e il tasto numerico 4 contemporaneamente il cursore diventerà di colore rosa.

Stampa in negativo

Il tasto CONTROL può essere usato per ottenere caratteri in negativo. In altre parole, il testo normale è nero su schermo bianco, mentre quando è in negativo diventa bianco su schermo nero. Potrai capire meglio se provi a digitare qualche carattere, poi tieni premuto il tasto CONTROL e premi il 9 (quello con la scritta sulla parte frontale RVS ON cioè inversione attivata). Lascia andare entrambi i tasti e prova a scrivere qualche carattere. Vedrai come appaiono invertiti i caratteri. Per tornare alla situazione normale tieni premuto il tasto CONTROL e premi la O (quello con la scritta sulla parte frontale RVS OFF cioè inversione disattivata).

Caratteri lampeggianti

Un'altra funzione del tasto CONTROL è di ottenere dei caratteri lampeggianti. Se tieni premuto il tasto CONTROL e premi il tasto con la virgola (che ha la scritta FLASH ON sulla parte frontale cioè lampeggio attivato) allora qualsiasi carattere che digiterai lampeggerà sullo schermo. Premendo il tasto CONTROL e il tasto del punto (che ha la scritta FLASH OFF sulla parte frontale cioè lampeggio disattivato) vedrai ritornare alla normalità gli altri caratteri digitati, ma i caratteri che stavano lampeggiando, continueranno a farlo.

I simboli grafici

Avrai probabilmente osservato che sulla parte frontale di molti dei tasti ci sono degli strani simboli fatti di linee e curve. Essi sono i simboli grafici e sono ottenibili con l'aiuto del tasto SHIFT e Commodore. Assicurati che il tuo computer sia inserito nel modo di scrittura maiuscolo poi premi il tasto SHIFT (o lo SHIFT

LOCK) e poi prova a premere alcuni dei tasti con i simboli grafici. Dovresti notare che il tasto SHIFT ti permette di ottenere i simboli grafici sulla parte frontale destra. Se rilasci lo SHIFT (o lo SHIFT LOCK) e tieni invece premuto il tasto Commodore mentre premi un tasto, allora otterrai il simbolo grafico sulla parte frontale sinistra.

Se ora cambi il modo di scrittura e torni alla minuscola, vedrai che alcuni dei simboli grafici sullo schermo si trasformano in lettere maiuscole. Questo succede perché quando sei nel modo di scrittura minuscolo e usi lo SHIFT, apparirà una lettera maiuscola invece di un simbolo grafico. In altre parole, quando sei nel modo di scrittura minuscolo non hai la possibilità di usare i simboli grafici sulla parte frontale destra del tasto e se usi la maiuscola con uno di questi tasti avrai solo una lettera maiuscola.

Il tasto RETURN

Quando volta durante i tuoi esperimenti con la tastiera, ti sarà capitato di premere il tasto RETURN, nel qual caso avrai ricevuto il messaggio '?SYNTAX ERROR' (ERRORE DI SINTASSI). Se ciò succede mentre stai facendo i tuoi esperimenti, non preoccuparti. Il tasto RETURN dice al computer di eseguire una istruzione che hai digitato, così se digiti la parola 'CIAO' e poi premi il tasto RETURN riceverai il messaggio '?SYNTAX ERROR' perché il computer non capisce la parola.

Ci sono ora solo sei tasti che ancora non conosci e non sai usare: i 4 tasti funzione (da F1 a F8), il tasto ESC e il tasto RUN/STOP. Verranno spiegati in seguito.

Il pulsante RESET

Sulla parte destra del computer c'è un piccolo pulsante chiamato "RESET" (azzeramento). Premendolo, il computer esegue una partenza a freddo, cioè torna alla situazione immediatamente seguente all'accensione e tutti i dati in memoria vengono persi. Comunque se tieni premuto il tasto RUN/STOP e premi questo pulsante, lo schermo si pulisce e appare il messaggio 'MONITOR' seguito da una serie di lettere e di numeri. Se ora provi a digitare la X e premi il tasto RETURN tutto tornerà normale e qualunque programma tu abbia avuto in memoria ci sarà ancora (partenza a caldo).

Riassunto

Ecco un breve riassunto sull'uso dei tasti speciali.

SHIFT: per ottenere simboli (come $, &, [), per ottenere i simboli grafici sulla parte destra del tasto e, usato in unione con il tasto Commodore, per passare dal modo di scrittura maiuscolo a quello minuscolo.

SHIFT LOCK: mantiene inserita la posizione SHIFT.

COMMODORE: per ottenere il colore in basso sul tasto (da 1 a 8), i simboli grafici sulla parte sinistra, ed è usato insieme al tasto SHIFT per passare dal modo di scrittura maiuscolo a quello minuscolo.

CLEAR/HOME: è usato per muovere il cursore alla sua posizione home (nell'angolo in alto a sinistra dello schermo). Quando SHIFT è premuto si usa per pulire lo schermo.

INST/DEL: è usato per cancellare caratteri. Quando SHIFT è premuto inserisce spazio tra i caratteri.

CONTROL: è usato per ottenere i colori in alto sui tasti numerici (da 1 a 8) e insieme ai tasti 9 e 0 per passare dalla stampa normale a quella in negativo e viceversa. Usato insieme alla virgola e al punto serve per far lampeggiare i caratteri e viceversa.

TASTI CON LE FRECCE: sono usati per muovere il cursore sullo schermo.

CAPITOLO 2

L'ISTRUZIONE PRINT

Ora dovresti aver preso abbastanza confidenza con la tastiera e sapere approssimativamente dove si trovano i tasti, per incominciare a far lavorare per te il computer.

Una delle possibilità principali di un computer e di saper fare i calcoli. Per poterli fare devi usare il comando PRINT.

Diciamo ad es. che vuoi risolvere l'addizione 9 + 7. Per farlo digita questo comando:

```
PRINT 9+7
```

e poi premi il tasto RETURN. Ciò dice al tuo computer di eseguire il comando che hai appena digitato, così il computer risolverà l'operazione 9 + 7 e mostrerà la risposta sullo schermo. Se apparirà un messaggio d'errore anziché la soluzione, avrai fatto un errore nella digitazione. In questo caso basterà muovere il cursore verso la parte in cui hai fatto l'errore, correggerlo (nello stesso modo in cui hai corretto la frase nel capitolo precedente) e premere il tasto RETURN.

Quando dai il comando sopracitato, dici al computer di stampare (PRINT) (o mostrare sullo schermo) la soluzione di 9 + 7. Naturalmente il tuo computer può eseguire altri calcoli oltre l'addizione. Prova le seguenti operazioni (ricorda di premere il tasto RETURN dopo ogni comando):

```
PRINT 32-17

PRINT 9*54

PRINT 99/11

PRINT 1564+1928
```

Avrai osservato da questi calcoli che il segno * significa 'moltiplicato per' e il segno / significa 'diviso per'. Questo tipo di notazione è comune a quasi tutti i computer. Prova a fare alcuni calcoli da solo e guarda i risultati.

Se hai provato a fare qualche calcolo multiplo (come 92 + 8/4) allora potresti aver pensato che il computer abbia fatto un errore nella sua risposta. Infatti prova questo calcolo:

```
PRINT 6+4/2
```

Sarai sorpreso di vedere che la risposta è 8 e non 5. Questo succede perché il computer dà sempre la priorità a certi calcoli rispetto ad altri. In questo caso il tuo computer (e molti altri) risolve prima le moltiplicazioni e le divisioni e poi le sottrazioni e le addizioni. Questo significa che il computer esegue la somma sopracitata come 6 + (la risposta di 4/2) che, naturalmente da 8. Per avere come soluzione 5 dovrai fare così:

```
PRINT (6+4)/2
```

Mettendo la parentesi tra '6 + 4' dici al computer di risolvere prima questa parte dell'operazione in modo che il totale (la soluzione di 6 + 4) sia diviso per 2 e dia 5.

Avrai notato che c'è una freccia verso l'alto sul tasto 0 (zero). Questo simbolo significa 'alla potenza di', così se digiti:

```
PRINT 3↑4
```

allora la soluzione 81 apparirà sullo schermo, perché 3 elevato alla quarta è 81 (3 × 3 × 3 × 3 fa 81).

Il calcolo esponenziale viene eseguito prima di ogni altro calcolo, così l'ordine di priorità è il seguente:

Elevamento a potenza, moltiplicazione o divisione, addizione o sottrazione.

Se per es. il computer deve eseguire l'operazione 5 + 7 – 2 eseguirà i calcoli nell'ordine in cui sono stati scritti, perché l'addizione e la sottrazione hanno priorità equivalente, come la moltiplicazione e la divisione. È possibile lavorare anche con i numeri negativi. Tutto quello che devi fare è mettere un segno meno davanti al numero per indicare che è negativo. Per es.:

```
PRINT -2+7
```

darà la soluzione 5.

Riassunto

L'istruzione PRINT deve essere usata per poter eseguire le operazioni con il formato PRINT (calcoli). Le operazioni vengono eseguite nel seguente ordine:

Elevamento a potenza;
Moltiplicazione e divisione;
Addizione e sottrazione.

Se una operazione contiene ad es. una sottrazione seguita da una addizione, verrà eseguita per prima la sottrazione perché i due tipi di calcolo hanno priorità equivalente.
Le parentesi possono essere utilizzate per una parte del calcolo se vuoi che il computer lo risolva per primo, senza tenere conto del tipo di calcolo.

Stampare messaggi

Una delle altre funzioni a cui può assolvere la istruzione PRINT è la stampa dei messaggi sullo schermo. Se digiti:

```
PRINT "CIAO, COME VA?"
```

(e naturalmente premi il tasto RETURN) vedrai apparire sullo schermo il messaggio CIAO, COME VA?
L'istruzione PRINT scriverà sullo schermo qualsiasi messaggio che metti tra virgolette. Le virgolette dicono al computer che ciò che è contenuto tra di esse, non è una operazione ma esattamente ciò che vuoi che appaia sullo schermo.

Cambiare il colore del testo

Può essere utile poter far apparire un messaggio con differenti colori. Per farlo devi includere il cambiamento di colore nell'istruzione PRINT che usi per far apparire questi messaggi. È facile da fare: devi solo digitare l'istruzione PRINT normalmente e poi, dove vuoi cambiare il colore, tenere premuto il tasto CONTROL o Commodore (dipende da quale colore desideri) e poi il tasto numerico con il colore che vuoi. Per es. se vuoi far apparire il messaggio CIAO LUIGI con la parola CIAO in rosso e la parola LUIGI in blu, devi digitare questo (dove vedi qualcosa tra parentesi significa che devi eseguire le istruzioni scritte, senza digitare né le parentesi quadre né la frase contenuta in esse).

PRINT "*[tieni premuto il tasto CONTROL e premi il tasto 3, poi lasciali entrambi]* CIAO *[tieni premuto il tasto CONTROL e premi il tasto 7, poi lasciali entrambi]* LUIGI"

Quando premi il RETURN vedrai apparire questo messaggio colorato sullo schermo. Come puoi vedere, quando cambi il colore durante l'istruzione PRINT, appare un carattere in negativo; ciò succede per ricordarti dove hai cambiato colore e quale hai scelto.

Abbreviazione per PRINT

Può essere utile sapere che, invece di digitare PRINT ogni volta, puoi usare il simbolo ? Il computer leggerà il segno ? come PRINT e lo esegue come se avessi usato il comando completo. Per es. ,?"CIAO" darà esattamente lo stesso risultato di PRINT "CIAO". Questo sistema risparmia molti caratteri, perciò cerca di ricordartelo.

Hai fatto ormai i primi passi per diventare un buon programmatore; sai già come far risolvere le operazioni al tuo computer (usando l'istruzione PRINT) e sai anche come far apparire i messaggi sullo schermo con differenti colori. Sai come usare la tastiera e sai anche che il computer non esegue nessun comando fino a quando non premi il tasto RETURN.

Riassunto

Tutto ciò che è all'interno delle virgolette dopo la istruzione PRINT apparirà sullo schermo così come è, tra virgolette.

Per cambiare il colore del testo durante l'istruzione PRINT si usa un sistema uguale a quello usato per cambiare normalmente il colore del testo. Un carattere in negativo apparirà per indicare dove hai cambiato colore, e cosa hai cambiato.

L'istruzione PRINT può essere abbreviata con il simbolo ?.

Sarebbe una buona idea se ti mettessi a fare degli esperimenti, almeno per mezz'ora, con tutto ciò che hai imparato fino ad ora e assicurarti di avere capito bene. Ti servirà per evitare di dover tornare indietro ogni 5 minuti, ed è importante perché presto scriverai il tuo primo programma in BASIC.

CAPITOLO 3

IL TUO PRIMO PROGRAMMA

Numeri di linea

Avrai certamente sentito parlare dei programmi di computer e ti sarai probabilmente chiesto cosa fossero esattamente. Un programma per computer è una serie di comandi che vengono eseguiti secondo un ordine stabilito dal computer. Fino ad ora hai detto al computer di eseguire soltanto il comando digitato. Questo sistema si chiama modo DIRETTO e il computer dimentica il comando dopo averlo eseguito, ma non è di gran utilità mentre scrivi un programma, perché di certo non vuoi che i comandi vengano eseguiti immediatamente. Per risolvere questo problema si usano i numeri di linea, che ti permettono di eseguire un programma più volte, tante quante desideri. I numeri di linea vengono usati per far ricordare al computer ogni comando che tu dai e per dirgli in quale ordine vuoi che questi comandi vengano eseguiti. La maggior parte delle persone preferisce avere i propri numeri di linea di 10 in 10 (ad es. 10, 20, 30...) in modo da avere spazio a disposizione per aggiungere linee in più, come vedrai tra poco.

Così eccoci al tuo primo programma. Osserverai che ogni comando è su una linea separata, con un proprio numero di linea. Prova a digitare questo programma esattamente come lo vedi (ricordati di premere il RETURN alla fine di ogni linea):

```
10 SCNCLR
20 PRINT"CIAO !!!"
30 PRINT"QUESTO E' IL TUO PRIMO PROGRAMMA PER COMPUTER ?"
40 PRINT"NON E' CERTO DI GRANDE EFFETTO,"
50 PRINT"MA I PROGRAMMI MIGLIORERANNO !"
```

Bene, non è successo molto, vero? Infatti abbiamo soltanto digitato un programma che il computer ricorderà fino a quando non gli diremo di dimenticarlo, ma per far funzionare il programma devi digitare il comando RUN (e premere il RETURN naturalmente). Appena l'avrai fatto, il computer eseguirà il programma.

Qualche volta potrai avere un messaggio d'errore, nel qual caso il computer ti dirà in quale linea c'è l'errore. Se succede, dovrai digitare di nuovo quella linea.

SCNCLR

Avrai osservato la presenza di un nuovo comando alla linea 10. Esso dice al computer di pulire lo schermo, esattamente come quando premi i tasti SHIFT e CLEAR/HOME. Tutto il resto del programma è costruito con l'istruzione PRINT, così potrai vedere cosa succede. In alternativa al comando SCNCLR puoi digitare un simbolo a cuore, ottenibile tenendo premuto SHIFT e poi il tasto CLEAR/HOME dopo aver digitato PRINT.

Così, il programma è inserito nella memoria ed è stato eseguito, ma cosa succede dopo? Bene, è possibile vedere il programma che hai scritto, digitando semplicemente LIST (senza dimenticare naturalmente di premere il RETURN). Questo comando farà apparire il programma sullo schermo linea dopo linea (sebbene ciò succeda abbastanza velocemente).

Avrai osservato che quando il programma sta girando (funzionando) ogni istruzione PRINT stampa il suo messaggio su linee separate. È possibile comunque avere due istruzioni PRINT consecutive. Ti sembrerà un po' confuso, ma aggiungi queste linee al programma per vedere come funziona:

```
60 PRINT "3232+5674=";
70 PRINT 3232+5674
```

Quando fai eseguire il programma vedrai lo stesso messaggio di prima, ma alla fine ci sarà una linea con '3232 + 5674 = 8906'.

La linea 60 del programma dice al computer di scrivere '3232 + 5674 =', e la linea 70 dice invece di dare al soluzione del calcolo 3232 + 5674 e scriverla sullo schermo. La cosa importante è sapere perché la seconda istruzione PRINT dà la soluzione sulla stessa linea in cui l'istruzione PRINT scrive l'operazione. Se guardi alla fine della linea 60 vedrai un punto e virgola (;): è questo segno che dice al computer di non andare a capo.

Ti ricorderai che all'inizio del capitolo avevo accennato al fatto che la maggior parte delle persone preferisce numerare le proprie linee di programma di 10 in 10, perché ciò permette di inserire linee in più anche dopo qualche tempo. Per vedere cosa significa, digita questa linea e poi digita LIST:

```
55 PRINT "UNA OPERAZIONE"
```

Quando il programma appare sullo schermo, vedrai che la nuova linea (linea 55) è stata inserita tra le linee 50 e 60. Prova a far eseguire il programma e vedi che effetto produce questa linea.

Se fai un errore digitando il programma, dovrai digitare di nuovo la linea sbagliata. Ciò ha l'effetto di correggere l'errore perché se dai un numero di linea uguale a quello di una linea che esiste già, il computer sostituirà la vecchia linea con quella nuova. Dato che potrebbe sembrare confuso all'inizio, prova a digitare questa linea:

```
55 PRINT "QUESTA LINEA E' STATA CAMBIATA!"
```

Quando digiti LIST vedrai che la vecchia linea numero 55 è stata sostituita da una nuova e cioè quella che hai appena digitato.

Cancellare una linea

È possibile anche cancellare una linea non necessaria di un programma. Per farlo devi semplicemente digitare il numero di linea che vuoi togliere e premere il tasto RETURN. Per es., digita il numero 55 e poi premi il RETURN: la linea 55 verrà tolta dal programma (digita LIST per esserne sicuro).

Linee con più istruzioni

Fino ad ora abbiamo avuto solo un comando per ogni linea. Il tuo computer però ti permette di avere più di un comando per ogni linea, infatti puoi avere tanti comandi quanti ne vuoi e di qualsiasi lunghezza, fino a non più di 80 caratteri (o due linee sullo schermo). Ci sono alcune eccezioni, ma verranno spiegate quando sarà il momento. Prova a cambiare la linea 10 con questo:

```
10 SCNCLR:PRINT "QUESTA E' UNA LINEA A PIU' ISTRUZIONI"
```

Ora la linea 10 ha due comandi, il comando SCNCLR e l'istruzione PRINT separati dai due punti (:). Tutto quello che devi fare per avere più di un comando sulla stessa linea, è separare ogni comando con i due punti (:).

Riassunto

Per far ricordare al computer una serie di istruzioni, devi dare dei numeri di linea. Ciò ti permette di dare l'ordine che vuoi per l'esecuzione delle istruzioni e ti permette anche di eseguire la serie di comandi (cioè il programma) quante volte vuoi.

I numeri di linea vanno di solito di 10 in 10 per permettere, più tardi, l'inserimento di linee in più. Non è essenziale ma è utile per aggiungere qualcosa in un programma.

Le linee di programma possono essere cancellate facilmente digitando il numero di linea e premendo poi il RETURN.

È possibile vedere il programma sullo schermo digitando LIST. Un programma viene eseguito digitando RUN.

Su una linea può essere inserita più di una istruzione, basta separare ogni istruzione con i due punti (:).

Potrebbe essere una buona idea se facessi delle prove con questo programma, o se provassi a scriverne uno da solo (in questo caso dovrai digitare NEW per liberarti del vecchio programma). Ti aiuterà a capire una delle parti più importanti della programmazione in BASIC.

Variabili numeriche

Le variabili sono estremamente importanti per un programmatore in BASIC e senza di esse sarebbe molto difficile scrivere un programma che fosse di qualche utilità.

Una variabile è un valore che può essere cambiato; ci sono tre tipi di variabili. Daremo prima una occhiata alle variabili numeriche.

Per rappresentare questi valori vengono usate delle lettere in modo che ad es. possiamo dire al computer che vogliamo la lettera A per simbolizzare il numero 34. Prova a digitare questo breve programma (devi prima digitare NEW e RETURN per liberarti di qualsiasi programma che fosse in memoria):

```
READY.

10 SCNCLR
20 A=34:PRINTA
```

Quando esegui questo programma vedrai lo schermo pulito e vedrai apparire il numero 34. Questo perché abbiamo detto al computer che vogliamo la variabile A rappresentata dal numero 34 e ogni volta che faremo riferimento alla variabile A (come abbiamo fatto dicendo al computer di PRINT A) tutto verrà eseguito come se avessimo fatto riferimento al numero 34. Naturalmente puoi dire al computer di indicare con la A qualsiasi altro numero.

Dato che una variabile può essere usata per rappresentare un numero, possiamo considerarla proprio come un numero; possiamo cioè aggiungere, dividere o fare qualsiasi cosa che normalmente facciamo con i numeri. Prova ad aggiungere al tuo programma le seguenti linee per illustrare questo concetto:

```
30 B=52:PRINT B
40 PRINT"34+52=";A+B
50 PRINT"34*52=";A*B
60 C=A-B
70 PRINTC
```

Ecco la spiegazione di ciò che sta facendo il programma.

Linea 10: Pulisce lo schermo.
Linea 20: Ricorda che la variabile A rappresenta il numero 34 e lo mostra sullo schermo.
Linea 30: Ricorda che la variabile B rappresenta il numero 52 e lo mostra sullo schermo.
Linea 40: Mostra il messaggio 34 + 52=, poi guarda cosa indicano le variabili A e B e le somma prima di mostrare la soluzione sulla stessa linea.
Linea 50: Mostra il messaggio 34 * 52 =, poi guarda cosa indicano le variabili A e B e le moltiplica prima di mostrare la soluzione sulla stessa linea.
Linea 60: Guarda cosa rappresentano le variabili A e B, poi sottrae A da B e ricorda che la variabile C rappresenta la soluzione.
Linea 70: Guarda cosa rappresenta la variabile C e mostra il numero sullo schermo.

Una variabile può essere una lettera qualsiasi o in pratica qualsiasi combinazione di lettere e numeri, così B, CIAO, LUIGI, A3, ZT99 e NUMERO9 possono essere usati come variabili. Comunque tutte le variabili devono iniziare con una lettera, così A9 può essere una variabile, mentre 9A no.

Una variabile non può essere usata se c'è un comando nella variabile stessa, così PRINTER non può essere usato (perché l'istruzione PRINT è un comando) né BRUNO (RUN) o LISTA (LIST).

Una variabile può essere di qualsiasi lunghezza (lunga tanto quanto una linea di programma), ma il computer riconosce solo le prime due lettere della variabile. Ciò significa che le variabili CIAO e CIMA vengono entrambe lette dal computer come CI e così sono praticamente uguali.

Il computer indica con zero tutte le variabili che non hai ancora usato, così se vuoi A = B, anche se non hai ancora usato la variabile B, il computer porrà la variabile A uguale a zero.

Infine abbiamo assegnato dei valori alle variabili scrivendo ad es., A = 1. C'è un comando in BASIC, chiamato LET, che può essere usato per dare dei valori alle variabili. Comunque non è necessario usarlo perché il computer legge sempre il comando LET A = 1 come A = 1, che è esattamente la formula che abbiamo usato fino ad ora per assegnare valori alle variabili. Se vedi il comando LET in un programma, puoi tralasciarlo.

Variabili intere (integer)

Le variabili a cui abbiamo fatto riferimento fino ad ora, possono rappresentare un qualsiasi numero, inclusi i numeri decimali. C'è un altro tipo di variabile, comunque, che può rappresentare solo numeri interi, cioè numeri senza decimali. Questo tipo di variabile viene chiamata variabile intera (integer).

Come le variabili normali, le variabili intere possono essere in pratica costituite da qualsiasi combinazione di lettere e numeri e valgono tutte le normali regole. Le variabili intere possono essere riconosciute dal segno di percentuale (%) che le segue. Per es. LUIGI% è una variabile intera, mentre LUIGI è una variabile normale o numerica.

Una variabile numerica e una intera possono avere lo stesso nome, possono contenere entrambe numeri differenti (sebbene la variabile intera possa contenere solo un intero).

Prova questo breve programma che illustra le differenze tra i due tipi di variabile (prima ricordati di digitare NEW):

```
10 SCNCLR
20 H=56.276:H%=56.276
30 PRINTH,H%
```

ATTENTO: per i computer la virgola dei decimali (,) è sostituita dal punto (.). Il punto che a volte si usa per le migliaia, i milioni, ecc... **DEVE** essere omesso. Perciò 10,4 diventa 10.4, 32.712,3 diventa 32712.3.

Questo programma pulisce lo schermo, poi assegna il valore 56.276 alla variabile H. Poi dà alla H% lo stesso valore di H, ma dato che H% è una variabile intera, tronca i decimali ed è pari a 56, come si può vedere quando la linea 30 mostra il valore di entrambe le variabili.

Avrai osservato che una virgola viene usata per separare le due variabili nella linea 30. Questa virgola stabilisce che il valore della variabile H% venga mostrato a partire dall'undicesimo spazio sullo schermo.

Variabili stringa

Un altro tipo di variabile è la variabile stringa. Essa può rappresentare lettere, numeri o altri caratteri ed è distinguibile dagli altri tipi di variabili dal simbolo del dollaro ($) detto stringa, che la segue.

Prova a digitare NEW e introdurre questo programma che usa variabili stringa:

```
10 SCNCLR
20 A$="CIAO":B$="A TUTTI"
30 PRINTA$;" ";B$
```

Come puoi vedere i caratteri che le variabili stringa possono rappresentare, devono essere tra virgolette.

Ci sono certe cose che non puoi fare con le variabili. Ecco alcuni esempi:

`A="LUIGI"`	(una variabile numerica non può rappresentare dei caratteri);
`PRINTER=3`	(una variabile non può incorporare un comando);
`RUNNER$="CIAO"`	(e non può farlo nemmeno una variabile stringa o una intera);
`B%=A$`	(una variabile numerica o intera non può rappresentare il contenuto di una variabile stringa e viceversa).

Riassunto

Ci sono tre tipi di variabili. Esse sono le variabili numeriche, intere e le variabili stringa.

Le v.n. sono rappresentate da una lettera o da un gruppo di lettere e numeri (es. Z, FD2, H34) e possono contenere solo numeri.

Le v.i. sono rappresentate come le variabili numeriche ma hanno un simbolo di percentuale che le segue (es. Z%, FD2%, H34%) e riconoscono solo gli interi (cioè se un numero decimale viene assegnato ad una v.i., la parte dopo la virgola verrà ignorata).

Le v.s. sono rappresentate come le variabili numeriche, ma hanno il simbolo del dollaro che le segue (es. Z$, FD2$, H34$) e possono contenere qualsiasi carattere. I caratteri devono essere inseriti tra virgolette.

Il comando LET può essere usato per assegnare dei valori alle variabili, ma non è necessario.

Ora è il momento giusto per fare degli esperimenti con le variabili prima di passare al prossimo capitolo. È importante che tu capisca come usarle, e un po' di tempo speso per assicurarti di averle capite bene, può risparmiarti ore di tentativi più tardi.

LIST

Hai già usato il comando LIST per far apparire il programma sullo schermo, ma il LIST può fare molto di più di quanto hai imparato. È possibile listare solo una parte del programma, per es. se vuoi vedere le linee dalla 30 alla 90 inclusa allora digiterai:

```
LIST 30-90
```

Se vuoi vedere tutto il programma fino alla linea 70 digiterai:

```
LIST -70
```

Potresti voler vedere solo le ultime linee del programma, la linea 100 e quelle seguenti per es., nel qual caso digiterai:

```
LIST 100-
```

Se vuoi vedere per es. solo la linea 25, digiterai semplicemente:

```
LIST 25
```

È possibile rallentare la velocità alla quale un programma sta listando, tenendo premuto il tasto Commodore. Se qualche volta hai bisogno di fermare un programma che sta listando, devi semplicemente premere il tasto RUN/STOP.

RUN

Hai già usato il comando RUN e comunque sai che è usato per far partire un programma dall'inizio. È anche possibile far partire un programma da una linea che non sia la prima. Per es. se vuoi far partire il programma dalla linea 50 in avanti digiterai:

```
RUN 50
```

Il comando RUN rimette a zero qualsiasi variabile numerica e svuota le variabili stringa.

NEW

È un altro comando che hai già usato. Esso cancella un programma dalla memoria ed allo stesso tempo azzera tutte le variabili (come per RUN). Devi fare attenzione quando usi questo comando perché una volta che l'hai usato il programma se ne è andato definitivamente.

INPUT

Il comando INPUT è molto utile nei programmi perché permette al computer di fare domande ad una persona e poi agire a seconda della risposta. Esso in realtà aspetta che la persona inserisca una risposta e poi la inserisce in una variabile. Ecco un programma che illustra il comando INPUT:

```
10 SCNCLR
20 PRINT"CIAO,COME TI CHIAMI";:INPUT NOME$
30 PRINT"SCRIVI UN NUMERO";:INPUT A
40 SCNCLR
50 PRINT"CIAO ";NOME$
60 PRINT"HAI SCRITTO IL NUMERO";A
```

Quando esegui questo programma lo schermo si pulirà e ti verrà chiesto di digitare il tuo nome (osserva il punto di domanda che appare — il computer lo fa apparire automaticamente quando sta eseguendo un comando INPUT e quindi aspetta una risposta). Non succederà niente fino a quando non avrai digitato il tuo nome e premuto il tasto RETURN. Appena l'avrai fatto, ti verrà chiesto di digitare un numero e non succederà niente fino a quando non lo farai (e premi il tasto RETURN naturalmente). Poi lo schermo si pulirà di nuovo e il computer ti dirà CIAO e ti dirà il numero che hai digitato.

Se guardi il programma precedente vedrai che il tuo nome è stato inserito nella variabile NOME$ e il numero che digiti è inserito per rappresentare la variabile A.

Ciò succede perché con INPUT dici al computer di aspettare qualcosa (numero o caratteri) e inserirlo nella variabile specificata dopo INPUT. In questo caso usiamo la variabile NOME$ per rappresentare il tuo nome e la variabile A per rappresentare il numero che hai digitato.

Senza dubbio avrai notato che nel precedente programma abbiamo usato una istruzione PRINT per fare la domanda. Possiamo comunque incorporare un breve messaggio nel comando INPUT. Per es., invece di avere due comandi come:

```
10 PRINT "QUANTO FA 27+49";:INPUT Z
```

Potremmo avere:

```
10 INPUT "QUANTO FA 27+49";Z
```

La seconda linea (con il messaggio come una parte del comando INPUT) mostrerà il messaggio QUANTO FA 27 + 49 sullo schermo e poi aspetterà una risposta. La soluzione è nella variabile Z.

Riassunto

Il comando INPUT attende che qualcosa venga inserito dalla tastiera e lo assegna ad una variabile. Può essere incorporato un messaggio nel comando INPUT per evitare di usare separatamente le istruzioni PRINT.

CAPITOLO 4

PROGRAMMAZIONE STRUTTURATA

IF ... THEN ... ELSE (SE .... ALLORA ... ALTRIMENTI)

Una delle utilizzazioni più importanti di un computer è quella di poter paragonare un valore con un altro. Ciò è essenziale in molti programmi ed è una caratteristica che permette al computer di raggiungere utili mete, come ad es. può fare attraverso la ricerca dell'indirizzo di una persona e del suo numero di telefono, paragonando ogni nome di una lista con il nome che sta cercando.

Ci sono tre comandi per dire al computer di paragonare un valore con un altro e sono IF, THEN e ELSE che vengono usati insieme. Ecco un esempio della loro utilizzazione:

```
IF Z=42 THEN K=9:ELSE K=3
```

Questo dice al computer che se (IF) la variabile Z è uguale a 42 allora deve eseguire ciò che segue THEN, cioè assegna 9 a K, altrimenti deve eseguire ciò che segue ELSE, cioè assegna 3 a K. Se il valore di Z è 42 allora la parte di ELSE della linea viene ignorata e il computer esegue l'istruzione o la linea seguente. Ci sono certi simboli che possono essere usati per paragonare un valore con un altro. Ci sono '>' (maggiore di), '<' (minore di) e '=' (uguale a) che possono essere ad es. combinati così > = cioè maggiore o uguale, e così via. Ecco alcuni esempi:

```
IF LUIGI=32 THEN GIOVANNI=30:ELSE GIOVANNI=22
```

Se il valore della variabile LUIGI è 32 allora assegna il valore 30 alla variabile GIOVANNI, altrimenti assegna il valore 22 alla variabile GIOVANNI.

```
IF A$<>"Y" THEN SCNCLR
```

Se la variabile stringa A$ non rappresenta la lettera Y, si pulisce lo schermo. Se A$ rappresenta la lettera Y allora viene eseguita la linea seguente.

```
IF KK3>99 THEN RUN
```

Se il valore della variabile KK3 è maggiore di 99 allora esegui il programma (RUN). Ci sono altri due operatori. Essi sono AND e OR (E e OPPURE) e vengono usati così:

```
IF A=1 AND B=1 THEN C=1
```

Se (IF) il valore della variabile 'A' è 1 e (AND) il valore della variabile B è 1 allora (THEN) assegna il valore 1 alla variabile C.

```
IF A=1 OR B=1 THEN C=1
```

Se (IF) il valore della variabile 'A' è 1 oppure (OR) il valore della variabile B è 1 (o entrambi sono 1) allora (THEN) assegna il valore 1 alla variabile C.

Qualsiasi comando che segue l'istruzione THEN verrà eseguito solo se le condizioni specificate vengono rispettate.

Ecco un programma esemplificativo che incorpora la maggior parte dei comandi che hai imparato fino ad ora:

```
10 SCNCLR
20 INPUT"CIAO,CI SIAMO GIA' CONOSCIUTI";IN$
30 IF IN$="S" OR IN$="SI" THEN PRINT"MI DISPI
   ACE MA NON MI RICORDO DI TE"
40 INPUT"QUAL'E' IL TUO NOME";NOME$
50 PRINT"PUOI CHIAMARMI COMMODORE !"
60 INPUT"TI DISPIACEREBBE SE TI CHIEDESSI QUA
   NTI ANNI HAI";QU$
70 IF QU$="S" OR QU$="SI" THEN PRINT"IN QUEST
   O CASO NON TE LO CHIEDO"
80 IF QU$="N" OR QU$="NO" THEN INPUT"PER FAVO
   RE DIMMI QUANTI ANNI HAI";ANNI$
90 PRINT
100 PRINT"BENE,MI DISPIACE MA ADESSO DEVO AND
    ARE."
110 PRINT"SPERO CHE CI INCONTREREMO ANCORA,";
    NOME$
```

E qui di seguito la spiegazione linea per linea.

Linea 10: Pulisce lo schermo.
Linea 20: Mostra il messaggio "CIAO CI SIAMO GIÀ CONOSCIUTI" e poi aspetta una risposta da assegnare alla variabile IN$.
Linea 30: Se il carattere assegnato alla variabile IN$ è S o SI allora appare il messaggio "MI DISPIACE MA NON MI RICORDO DI TE".
Linea 40: Appare il messaggio "QUAL È IL TUO NOME" e aspetta una risposta che deve essere assegnata alla variabile NOME$.
Linea 50: Appare il messaggio "PUOI CHIAMARMI COMMODORE!".

Linea 60: Appare il messaggio "TI DISPIACEREBBE SE TI CHIEDESSI QUANTI ANNÍ HAI" e poi aspetta la risposta che deve essere assegnata alla variabile QU$.
Linea 70: Se il carattere assegnato alla variabile QU$ è S o SI allora appare il messaggio "IN QUESTO CASO NON TE LO CHIEDO".
Linea 80: Se il carattere assegnato alla variabile QU$ è N o No appare il messaggio "PER FAVORE DIMMI LA TUA ETÀ" e aspetta la risposta che deve essere assegnata alla variabile ANNI$.
Linea 90: Appare una linea vuota.
Linea 100: Appare il messaggio "BENE, MI DISPIACE MA ADESSO DEVO ANDARE".
Linea 110: Appare il messaggio "SPERO CHE CI INCONTREREMO ANCORA,"; NOME$.

Riassunto

La struttura IF ... THEN ... ELSE è usata per comparare un valore con un altro e può comparare sia tutti i tipi di variabile che tutti i numeri fissi o i caratteri (ad es. IF A$ = "Y", IF ZZ = 22).

Alcuni operatori speciali vengono usati con IF ... THEN ... ELSE per comparare degli elementi. Questi sono '>' (maggiore di), '<' (minore di) e '=' (uguale a) e possono essere combinati così: > = cioè maggiore di o uguale a.

AND e OR possono essere usati in modo che più di una condizione possa essere soddisfatta — IF A = 2 AND B = 1 THEN PRINT "CIAO" — il computer farà apparire CIAO solo se la variabile A rappresenta il valore 2 e la variabile B è uguale a 1.

Cicli FOR ... NEXT

Spesso è necessario eseguire le stesse istruzioni per parecchie volte. Ad es. potresti usare un programma come questo per indicare i multipli di 15 fino a 10 * 15:

```
10 SCNCLR
20 PRINT 1*15
30 PRINT 2*15
40 PRINT 3*15
50 PRINT 4*15
60 PRINT 5*15
70 PRINT 6*15
80 PRINT 7*15
90 PRINT 8*15
100 PRINT 9*15
110 PRINT 10*15
```

Digitare un tale programma, anche se funziona perfettamente, è un po' noioso e utilizza della memoria del computer. Per aiutarti quando vuoi eseguire una serie di istruzioni per parecchie volte, ci sono due comandi chiamati FOR e NEXT. Essi funzionano insieme così:

```
10 FOR N=1 TO 10
20 PRINT N*15
30 NEXT N
```

Questo breve programma costituito da solo tre righe, sostituisce il primo programma con le sue dieci istruzioni PRINT. Ecco come funziona il programma:

Linea 10: Dice al computer di incominciare a ripetere tutte le istruzioni tra il comando FOR e il NEXT 10 volte.
Linea 20: Moltiplica il valore corrente rappresentato dalla variabile N per 15 e mostra il risultato sullo schermo.
Linea 30: Se il valore rappresentato dalla variabile N è meno di 10 allora il valore di N viene aumentato di uno e il programma torna indietro alla linea 20.

Se ora provi a cambiare la linea 10 con:

```
10 FOR N=20 TO 30
```

e fai eseguire il programma, allora vedrai i multipli di 15 da 20 * 15 fino a 30 * 15. Ciò succede perché abbiamo detto al computer che volevamo il valore della variabile N a partire da 20 e aumentato di uno ogni volta fino a quando il ciclo arriva al valore di N = 30.

Un altro comando STEP, può essere aggiunto alla fine del comando FOR. Con esso dici al computer quanto vuoi che aggiunga alla variabile del ciclo ogni volta. Per es. se cambi la linea 10 con:

```
10 FOR N=20 TO 30 STEP 2
```

e fai eseguire il programma, vedrai i multipli di 15 da 20 a 30.

Puoi perfino andare all'indietro con un ciclo usando uno STEP negativo. Se cambi la linea 10 con:

```
10 FOR N=30 TO 20 STEP -1
```

e fai eseguire il programma, vedrai i multipli di 15 apparire sullo schermo, ma questa volta partiranno da 30 * 15 e andranno all'indietro fino a 20 * 15.

I cicli FOR ... NEXT possono essere 'nidificati' (messi uno nell'altro). Prova questo programma esemplificativo:

```
10 FOR N=1 TO 10
20 FOR M=1 TO 500:NEXT M
30 PRINT N
40 NEXT N
```

Quando esegui questo programma vedrai i numeri da uno a dieci apparire molto lentamente, perché abbiamo un ciclo nidificato sulla linea 20 che continua a girare 500 volte senza fare niente. Tutto ciò rallenta il programma in modo che il ciclo più esterno esegue le sue funzioni lentamente. Ecco un altro esempio di cicli nidificati:

```
10 SCNCLR
20 FOR N=1 TO 12
30 PRINT"LA TABELLINA DEL";N
40 FOR M=1 TO 12
50 IF M<10 THEN PRINT" ";
60 PRINT M;"X";N;"=";M*N
70 NEXT M
80 FOR Z=1 TO 2000:NEXT Z
90 SCNCLR
100 NEXT N
110 RUN
```

Questo programma contiene due cicli nidificati, uno dalla linea 40 alla 70 e l'altro sulla linea 80. Il ciclo principale, dalla linea 20 alla 100 sceglie quale tabellina deve apparire. Parte da 1 e aumenta ogni volta di uno fino a quando la variabile N arriva a 12.

Il primo ciclo nidificato usa la variabile M e si estende da 1 a 12. La linea 50 mostra uno spazio se il valore di M è meno di 10, per fare in modo che i numeri siano ben incolonnati — puoi vedere esattamente cosa fa, cancellando questa linea e facendo eseguire il programma. Il valore della M è moltiplicato per il valore di N ogni volta, in modo che appaiano sullo schermo i multipli di N da 1 a 12.

Il secondo ciclo nidificato sulla linea 80, usa la variabile Z ed è un ciclo di ritardo. Esso conta fino a 2000 per darti abbastanza tempo per leggere cosa c'è sullo schermo.

Lo schermo viene pulito per ogni tabellina dal comando SCNCLR sulla linea 90. Una volta che il programma ha finito, riparte con il comando RUN alla linea 110.

Riassunto

I cicli FOR ... NEXT vengono usati per eseguire un gruppo di istruzioni per parecchie volte. Il valore della variabile usata nel ciclo viene aumentata di uno ogni volta che raggiunge l'istruzione NEXT.

Il comando STEP può essere aggiunto al comando FOR per cambiare la quantità aggiunta alla variabile al raggiungimento di NEXT. Ad es. STEP 3 aggiungerebbe 3 ogni volta alla variabile e STEP –2 sottrarrebbe 2 dalla variabile.
Il comando NEXT stabilisce sempre la fine di un ciclo. Il nome della variabile può essere aggiunto (ad es. NEXT N), ma non è essenziale.

GOTO

Come saprai, quando il computer esegue un programma, lavora seguendo l'ordine numerico delle linee. È possibile comunque cambiare questo ordine, in modo che il computer possa, per es., eseguire la linea 10 e poi eseguire dalla linea 100 alla 150 e poi ritornare alla linea 20 e continuare da quel punto.
Il comando GOTO dice al computer di andare ad una linea ed eseguire il programma da quel punto. Prova con questo breve programma:

```
10 PRINT "#";
20 GOTO 10
```

Se fai eseguire questo programma allora vedrai il DIESIS o CANCELLINO (#) stampato sullo schermo continuamente. Per fermare il programma che si rivela senza fine devi premere il tasto RUN/STOP.
Il programma continua ad andare avanti perché il CANCELLINO è stato scritto dalla linea 10; la linea 20 dice al computer di andare alla linea 10, così viene stampato un altro CANCELLINO e il computer torna indietro alla linea 10 e così di seguito.

GOSUB e RETURN

Il comando GOSUB è molto simile al GOTO. Dice al computer di andare verso una 'SUBROUTINE' (sottoprocedura, sottoprogramma) e continuare il programma da quel punto. Se il computer allora trova un comando RETURN continuerà con il programma dal comando successivo al comando GOSUB.
Una 'SUBROUTINE' è un programma dentro al programma, in altre parole è un breve pezzo di programma che esegue alcune operazioni e può essere usato più volte. Così, potendole richiamare, si risparmia sia tempo che memoria, dato che esse necessitano di essere digitate solo una alla volta.
Ecco un breve programma che illustra il comando GOSUB e RETURN:

```
10 PRINT"ROUTINE UNO"
20 PRINT"***********"
30 GOSUB 50
40 GOTO10
50 PRINT"ROUTINE DUE"
60 PRINT"###########"
70 RETURN
```

Quando esegui questo programma vedrai apparire sullo schermo:

```
ROUTINE UNO
***********
ROUTINE DUE
###########
ROUTINE UNO
***********
ROUTINE DUE
###########
```

e questo verrà scritto più volte fino a quando non fermerai il programma.
Il programma funziona così:

Linea 10: Appare il messaggio 'ROUTINE UNO'.
Linea 20: Appare il messaggio ***********.
Linea 30: Va al sottoprogramma che inizia alla linea 50.
Linea 40: Va alla linea 10 e fa continuare il programma da lì.
Linea 50: Appare il messaggio 'ROUTINE DUE'.
Linea 60: Appare il messaggio '###########'.
Linea 70: Ritorna al comando immediatamente dopo il comando GOSUB che ha chiamato questa subroutine (in questo caso il computer andrà alla linea 40 siccome questa contiene il primo comando dopo il comando GOSUB).

Avrai capito che se una linea del genere appare in un programma:

```
100 GOTO 50:PRINT "CIAO"
```

allora l'istruzione PRINT non verrà mai eseguita, perché appena il computer raggiunge il comando GOTO andrà alla linea 50 e continuerà con il programma da quel punto. Comunque se la linea 100 è:

```
100 GOSUB 50:PRINT "CIAO"
```

allora l'istruzione PRINT verrà eseguita appena al computer viene detto di ritornare dalla subroutine sulla linea 50, dato che l'istruzione PRINT è il comando successivo al GOSUB.

Una cosa che non devi fare con un comando GOTO e GOSUB è di sostituire un numero di linea con una variabile, perché se vuoi andare (GOTO) alla linea 10 non puoi sostituire il 10 con una variabile, anche se essa ha come valore 10.

Il GOTO e GOSUB possono, naturalmente, essere usati con la struttura IF ... THEN ... ELSE, ottenendo buoni risultati.

Riassunto

Il comando GOTO dice al computer di andare verso un certo numero di linea e continuare con il programma da quel punto.

Il comando GOSUB dice al computer di andare verso una sottoprocedura partendo da un certo numero di linea e continuando con il programma da quel punto. Quando raggiunge un comando RETURN, il computer ritornerà all'istruzione immediatamente successiva all'istruzione GOSUB.

Ogni comando dopo il GOTO sulla stessa linea non viene eseguito.

Il numero di linea dopo il comando non può essere sostituito da una variabile.

Editare i programmi

Fino ad ora ogni volta che hai fatto un errore nei tuoi programmi, hai dovuto digitare di nuovo la linea. È possibile comunque alterare un programma senza far questo. Digita questo programma esattamente come è:

```
10 SCMCML
20 PRIINT "CIAOO"
30 GTO 20
```

Ovviamente questo programma è pieno di errori ma non sarebbe conveniente doverlo riscrivere. Fortunatamente possiamo invece editare il programma.

Muovi il cursore verso la M di SCMCLR e poi digita la N e poi RETURN: la linea 10 è stata corretta.

L'errore seguente è nella linea 20. Ci sono due errori e per correggerli dovresti prima muovere il cursore sopra la N di PRINT e poi premere il tasto INST/DEL; poi muovi il cursore sopra la seconda O di CIAOO e premi il tasto INST/DEL ancora. Premi il RETURN e la linea 20 è corretta.

Il comando GOTO nella linea 30 manca di una O. Per inserire questa lettera dovresti muovere il cursore verso la T e poi premere SHIFT e INST/DEL. Si aprirà uno spazio sufficiente per digitare la O al posto giusto. Dovresti poi premere il RETURN, digitare LIST e il tuo programma avrà una forma corretta.

Se decidi, mentre stai editando una linea, che vuoi lasciarla così come era, puoi premere SHIFT CLEAR/HOME (che pulisce lo schermo) oppure premere SHIFT e RETURN. Potresti anche muovere il cursore oltre la linea che vuoi correggere, perché nessun cambiamento è permanente fino a quando non si preme il tasto RETURN.

Registrare i programmi su nastro

Se volessi riutilizzare uno dei programmi che hai scritto precedentemente, avrai probabilmente considerato una seccatura doverlo digitare di nuovo ogni volta. Per evitare ciò puoi immagazzinare i programmi su un normale nastro (cassetta).

Avrai probabilmente un registratore Commodore 1531. Se non ce l'hai sarebbe una buona idea comperarlo, anche se vuoi usare un disk drive, perché un certo numero di programmi sono disponibili solo su cassetta. Il registratore deve essere collegato al connettore con l'indicazione CASSETTE sulla parte posteriore del computer (devi spegnere il computer mentre fai questa operazione). Tutto quello che ti serve è qualche nastro vergine. La maggior parte delle cassette di musica funzionano perfettamente se non sono più lunghe di 45 min, ma puoi anche comperare delle cassette speciali per computer.

Una volta che tutto è collegato e c'è una cassetta adatta nel registratore, devi far riavvolgere il nastro fino all'inizio. Ora sei pronto per salvare un programma. Questo programma andrà bene per fare una prova:

```
10 SCNCLR
20 PRINT"QUESTO PROGRAMMA E' STATO SALVATO"
30 PRINT"SU NASTRO E CARICATO CON SUCCESSO"
40 FOR N=1 TO 25
50 PRINT"***************************************
   ****";
60 NEXT N
70 FOR N=1 TO 2000:NEXT N
80 RUN
```

Ora tutto quello che devi fare è digitare:

```
SAVE "PROGRAMMA"
```

ti verrà chiesto di PREMERE IL TASTO PLAY E REGISTRARE SU NASTRO e dovrai agire proprio come quando vuoi registrare su un normale registratore a cassetta. Lo schermo diventerà bianco e la registrazione inizia. Dopo poco tempo lo schermo ritornerà normale e il registratore si ferma: il tuo programma è stato registrato su nastro.

Il comando SAVE "PROGRAMMA" dice al computer di fare una copia del programma che c'è in memoria sul nastro e dargli il nome di PROGRAMMA (dando al programma un nome facile da trovare più tardi). Come puoi vedere, il nome deve essere tra virgolette.

Verificare il programma

Puoi assicurarti che il programma sia stato registrato correttamente usando il comando VERIFY. Riavvolgi il nastro e digita:

```
VERIFY "PROGRAMMA"
```

Il computer ti dirà di PREMERE IL TASTO PLAY SUL REGISTRATORE e appena l'avrai fatto, lo schermo diventerà bianco e il registratore partirà. Il computer cercherà il programma che è stato salvato sul nastro con quel nome. Quando lo trova apparirà il messaggio:

```
FOUND PROGRAMMA        (TROVATO PROGRAMMA)
VERIFYING              (VERIFICA)
```

sullo schermo e poi si fermerà. A questo punto puoi risparmiare tempo premendo il tasto Commodore che dice al computer di non aspettare. Il computer procederà allora alla comparazione del programma che è su nastro con quello che è ancora in memoria. Se sono uguali (come dovrebbero essere perché hai ancora in memoria il programma che hai salvato) allora lo schermo ritornerà normale e il computer ti informerà che tutto va bene. Se il programma non è stato salvato correttamente, allora apparirà il messaggio:

```
VERIFY ERROR           (ERRORE DI VERIFICA)
```

Se ciò succede dovresti provare a riavvolgere il nastro, pulirlo e partire di nuovo. Se continui a non riuscire a salvare il programma, allora prova con un altro nastro. Se dopo alcuni nastri differenti non riesci ancora a salvare con successo un programma, dovresti portare il tuo computer e il registratore dal rivenditore Commodore e farteli controllare.

Caricare il programma

Ora, spero che tu sia riuscito a salvare il tuo programma sul nastro. Digita NEW e poi:

```
LOAD "PROGRAMMA"
```

Il computer ti chiederà di PREMERE IL TASTO PLAY SUL REGISTRATORE, come aveva fatto quando volevi verificare, e appena lo avrai fatto lo schermo diventerà bianco e il computer incomincerà a cercare il programma chiamato PROGRAMMA. Appena lo avrà trovato, apparirà il messaggio:

FOUND "PROGRAMMA"	(TROVATO PROGRAMMA)
LOADING	(CARICAMENTO)

Ci sarà ancora una pausa, che può essere accorciata premendo il tasto Commodore. Lo schermo diventerà bianco mentre il computer carica il programma. Quando appare il messaggio OK (tutto bene) puoi digitare LIST o RUN e il tuo programma apparirà.

Il nome dopo i comandi LOAD, SAVE e VERIFY non è obbligatorio, anche se è una buona idea dare un nome ad ogni programma. Per es. se hai appena digitato LOAD e premi il RETURN, il computer caricherà il primo programma che incontra in memoria.

Puoi aggiungere due numeri in più alla fine dei comandi LOAD e SAVE. Il primo indica il numero del dispositivo. Per es. il comando:

```
SAVE "GRAPH",8,1
```

salverà il programma in memoria sul dischetto, che è il dispositivo numero 8 e gli darà il nome di GRAPH. Per una spiegazione dei differenti numeri di dispositivi, fai riferimento al tuo Manuale. Aggiungendo ',1' dici al computer di registrare un segnalatore di fine nastro alla fine del programma. Ciò significa che se il computer sta cercando un programma sul nastro e incontra un segnalatore di fine nastro (End Of Tape = EOT) pensa di essere arrivato alla fine del nastro e viene perciò dato il messaggio FILE NOT FOUND ERROR.

Similmente il comando:

```
LOAD "GRAPH",1,1
```

farà in modo che il computer carichi il programma con il nome GRAPH dal nastro (il primo ',1' dice al computer di caricare dal nastro) e caricarlo nella memoria da dove era stato salvato. Aggiungendo ',1' alla fine del comando LOAD fai in modo che il computer carichi il programma nella stessa area di memoria da cui era stato preso per essere salvato. Se l'1 viene tralasciato il computer caricherà il programma all'inizio della memoria BASIC. Avrai bisogno di usare questo comando solo per caricare programmi in linguaggio macchina, che verranno spiegati più avanti, e che qualche volta vengono immagazzinati in differenti aree di memoria.

Riassunto

Il comando SAVE è usato per fare una copia del programma in memoria sul nastro. Si può dare un nome al programma per facilitare più tardi, la ricerca sul nastro.

Il nome del programma può essere un nome qualsiasi, ma non più lungo di 16 caratteri. Deve essere tra virgolette.

Il comando VERIFY è usato per controllare se il programma salvato su nastro è esattamente uguale a quello in memoria. Può essere usato per trovare la fine del programma semplicemente digitando VERIFY "nome del programma" e aspettare fino a quando il computer ti dà un errore di verifica.

Il comando LOAD è usato per leggere un programma dal nastro. Il nome del programma può essere usato, ma non è essenziale, se sai esattamente dov'è il programma sul nastro.

Alcuni programmi in linguaggio macchina devono essere caricati con LOAD "nome del programma" ,1,1.

CAPITOLO 5

QUALCHE IDEA IN PIÙ

INT

Il comando INT è usato per arrotondare i numeri. Per es. se digiti:

```
PRINT INT(32.3)
```

allora vedrai il numero 32 apparire sullo schermo, perché il numero 32.3 arrotondando diventa 32. INT arrotonderà sempre un numero per difetto, così 98.1 diventerà 98,–54.87 diventerà –55 e così via.

RND

Il comando RND, come INT è una 'funzione', cioè un comando che prende un numero e fa qualcosa con esso, prima di dare il risultato.

La funzione RND sceglie un numero decimale casuale tra 0 e 1. Il modo in cui questo numero viene scelto dipende dall'argomento che scegliamo. Questo argomento è tra parentesi, segue la funzione RND e può anche essere un numero negativo, un numero positivo o uno zero.

Se usiamo un argomento positivo, come:

```
PRINT RND (1)
```

allora il computer sceglierà un numero a caso che si ripeterà dopo parecchie migliaia di volte. Un argomento negativo come questo:

```
PRINT RND (-1)
```

sceglie l'inizio della serie di numeri. Ogni volta che viene richiesto in quella serie. Usando un argomento negativo si altera la posizione che quel numero aveva preso nell'elenco. Nel sopracitato esempio dovresti ottenere la risposta 2.99196472E–08, ma se usi un altro argomento negativo, ad es. –2 otterrai un numero differente.

Alla fine se usiamo zero come argomento, come questo:

```
PRINT RND (0)
```

allora il numero casuale viene scelto a seconda dell'orologio interno. Questo metodo di scelta di un numero a caso è probabilmente il migliore per molti scopi perché è praticamente impossibile che la sequenza venga ripetuta.

Questo programma sceglie 30 numeri casuali tra 1 e 10, di cui ciascuna decina con un argomento negativo, nullo o positivo.

```
10 SCNCLR
20 PRINT "ARGOMENTO POSITIVO":X=1
30 GOSUB 80
40 PRINT"ARGOMENTO NEGATIVO":X=-1
50 GOSUB 80
60 PRINT"ARGOMENTO ZERO":X=0
70 GOSUB 80:END
80 FOR N=1 TO 10
90 PRINT INT(RND(X)*9)+1;
100 NEXT N
110 FOR N=1 TO 4000:NEXT N
120 SCNCLR:RETURN
```

Osserverai che il numero casuale scelto nella linea 90 viene moltiplicato per nove e poi arrotondato per difetto. Questo ha l'effetto di scegliere un numero intero tra zero e nove. Aggiungendo uno al risultato è come se si scegliesse un numero tra uno e dieci.

CHAR

Il comando CHAR è molto simile al comando PRINT perché permette di mostrare i caratteri sullo schermo. CHAR differisce da PRINT solo perché permette di scegliere esattamente dove vogliamo che il carattere appaia sullo schermo.

Avrai già notato che puoi avere 40 caratteri in orizzontale e 25 in verticale. Ciò significa che puoi avere fino a 1000 caratteri sullo schermo.

Ora immagina che sia stata disegnata una rete sullo schermo della televisione e che ci siano 40 quadrati in orizzontale e 25 in verticale. Oltre la linea finale dei quadrati vengono scritti dei numeri da 0 a 39 e alla sinistra della prima colonna a sinistra dei quadrati, vengano scritti i numeri da 0 a 24, proprio come su una mappa. È possibile ora dare ad ogni quadrato sullo schermo delle coordinate, in modo che tu possa localizzare ogni carattere sullo schermo e dire esattamente dove si trova. Per es. se c'è una lettera nell'angolo in alto a sinistra dello schermo

è al punto 0,0 (0 orizzontale e 0 verticale). Se c'è una lettera nell'angolo in basso a destra sullo schermo allora sarà il punto 39,24 (39 orizzontale e 24 verticale).

Ora che sai come è diviso lo schermo e come ogni posizione abbia delle coordinate, puoi incominciare ad usare il comando CHAR. Tutto quello che devi fare è dire al computer in quale posizione vuoi che appaia il primo carattere del tuo messaggio e poi dargli il messaggio vero e proprio. Prova questo programma esemplificativo:

```
10 SCNCLR:INPUT"COORDINATA X";X
20 INPUT"COORDINATA Y";Y
30 CHAR 1,X,Y,"CIAO"
40 FOR N=1 TO 4000:NEXT N
50 RUN
```

Quando lo esegui ti verrà chiesta la coordinata X, in altre parole il numero della posizione orizzontale da cui vuoi che parta il messaggio. Ti verrà chiesta poi la coordinata Y ovvero a quale posizione verticale vuoi che parta il messaggio. Troverai probabilmente difficoltoso capire tutto questo all'inizio, perciò prova a fare degli esperimenti con X e Y differenti per vedere che risultati ottieni. Una volta che hai fatto ciò, avrai la parola CIAO sullo schermo, partendo dalla posizione che hai appena scelto.

Se guardi alla linea 30 del programma vedrai il comando CHAR. Sceglierai dove mettere le coordinate X e Y e il messaggio, ma c'è anche un numero subito dopo il comando CHAR. Questo numero dice al computer che vuoi che i caratteri appaiano nello stesso colore del testo. Cambiando questo numero con uno 0 si fanno apparire i caratteri dello stesso colore dello sfondo, e così non li vedrai.

Come puoi vedere l'1, le coordinate X e Y e il messaggio sono tutti separati tra di loro da una virgola. Il messaggio può essere di qualunque lunghezza, ma il comando deve stare in una sola linea di 80 caratteri (due linee di schermo).

Se provi a cambiare la linea 30 così:

```
30 CHAR 1,X,Y,"CIAO",1
```

e fai eseguire il programma, vedrai la parola CIAO apparire in negativo. Si può ottenere aggiungendo un 1 dopo il messaggio. Se non aggiungi niente o uno 0 i caratteri rimangono normali.

Riassunto

Lo schermo è formato da 1000 quadrati — 40 orizzontali e 25 verticali — ciò permette di dare ad ogni quadrato una coordinata.

Il comando CHAR viene usato per permettere di scegliere in che posizione sullo schermo vuoi che appaiano i caratteri o i messaggi.

Il messaggio è tra virgolette, proprio come con l'istruzione PRINT.
Aggiungendo '1' dopo il messaggio, fa apparire il messaggio in negativo.
I caratteri possono apparire con il colore del testo (CHAR 1) o con il colore dello sfondo (CHAR 0).

Mappa dello schermo

Ecco una mappa che mostra le coordinate di ogni posizione di un carattere sullo schermo. Per es. la posizione del carattere 10 verticale e 4 orizzontale ha le coordinate 9,3.

00 02 04 06 08 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38

00
02
04
06
08
10
12
14
16
18
20
22
24

Uso dei joysticks

I joysticks sono dei pezzi mobili dell'apparecchiatura e possono essere usati per svariati scopi, sebbene il più ovvio e il più comune sia per giocare ai video giochi. Puoi collegare due joysticks al tuo computer inserendo i cavetti negli spinotti contrassegnati da JOY 0 e JOY 1.
Utilizzarli è molto semplice: basta usare le funzioni del comando, che ti darà certi risultati a seconda della posizione del joystick.



Così se il joystick che stai usando si sposta verso l'alto la funzione del comando sarà di valore 1, se si sposta verso il basso a destra, il valore sarà 4. Se il pulsante per il fuoco è premuto allora 128 verrà aggiunto al numero direzionale.

Ora che sai quali risultati ottieni con i joysticks, sarebbe utile imparare come utilizzarli. Prova a digitare questo programma:

```
10 SCNCLR
20 PRINT"JOYSTICK DESTRO   JOYSTICK SINISTRO"
30 DE=JOY(1):SI=JOY(2)
40 CHAR 1,5,3,STR$(DE):CHAR 1,25,3,STR$(SI)
50 GOTO 30
```

Il joystick a destra corrisponde alla funzione JOY (1) e quello a sinistra alla funzione JOY (2) (questo può trarti in confusione dato che gli spinotti per i joysticks sono contrassegnati da JOY 0 e JOY 1).

Questo programma considera il valore del joystick destro corrispondente alla variabile DE e il valore del joystick sinistro alla variabile SI. Questi valori appaiono ora sullo schermo. Se il pulsante del fuoco è stato premuto su uno dei due joysticks, allora il numero direzionale viene aumentato di 128, come vedrai se proverai a premere il pulsante del fuoco. Quando il joystick si trova nella posizione centrale il valore che apparirà sarà zero.

GET

Conosci già un modo per ottenere informazioni tramite la tastiera ed è usando il comando INPUT. Comunque, puoi immaginarti come sarebbe se, giocando ad un video gioco, dovessi premere il RETURN dopo ogni mossa? È quello che faremmo se usassimo il comando INPUT.

Fortunatamente per noi, abbiamo due comandi che possono essere usati per verificare se un singolo tasto è stato premuto. Essi sono GET e GETKEY. Prova questo breve programma:

```
100 SCNCLR
110 PRINT"UN'ALTRA VOLTA (S/N) ?";
120 GET A$:IF A$="" THEN GOTO 120
130 IF A$="N" THEN END
140 IF A$="S" THEN RUN
150 GOTO 120
```

Quando esegui questo programma ti verrà chiesto UN'ALTRA VOLTA? (S/N) e non succederà niente fino a quando non premerai il tasto S o N. Se premi il tasto N il programma si fermerà, ma se premi il tasto S il programma girerà ancora. Se premi qualsiasi altro tasto non succederà niente.

La linea 120 contiene il comando GET. Come puoi vedere, con GET deve essere usata una variabile (in questo caso A$) perché il comando GET esamina la tastiera per vedere se qualche tasto è stato premuto e se ce n'è uno, allora il carattere del tasto viene attribuito alla variabile. Se nessuno dei tasti è stato premuto allora la variabile rimarrà vuota. La linea 120 verifica se A$ è vuota e se lo è, la linea 120 viene eseguita più volte fino a quando non trova un tasto premuto.

Le linee 130 e 140 verificano quale tasto è stato premuto. Se non è stata premuta né la S né la N allora non succede niente e la linea 150 dice al computer di tornare indietro alla linea 120 per ripetere.

Il comando GET non fa fermare il programma come invece fa il comando INPUT e può leggere solo un carattere per volta. Ciò significa che il programma continuerà indipendentemente dal fatto che un tasto sia premuto o no.

GETKEY

Il comando GETKEY è simile al comando GET, ma in questo caso aspetta che un tasto sia premuto e il programma non continuerà fino a quando non l'avrai fatto. Con questo sistema si evita di far verificare se c'è qualcosa nella variabile stringa che noi usiamo, ma ha lo svantaggio di far fermare il programma. Usando il comando GETKEY possiamo cambiare la linea 120 così:

```
120 GETKEY A$
```

Il programma funziona come in precedenza, ma non dobbiamo verificare più per vedere se la variabile A$ rappresenta qualcosa, perché il GETKEY aspetta fino a quando non viene premuto un tasto prima di assegnare il carattere di quel tasto alla variabile A$.

Riassunto

Il comando GET esamina la tastiera e assegna il carattere corrispondente a qualsiasi tasto sia stato premuto alla variabile stringa. Se non viene premuto alcun tasto la variabile rimarrà vuota.

Il comando GET non ferma il programma, a differenza del comando INPUT, e non ha bisogno che sia premuto il tasto RETURN.

Il comando GETKEY è simile al comando GET, ma ferma il programma fino a quando non viene premuto un tasto.

Cicli DO ... LOOP

Abbiamo già incontrato una forma di ciclo, quello FOR ... NEXT, ma è usato meno frequentemente. Ecco un esempio di DO ... LOOP:

```
10 DO
20 PRINT "CIAO"
30 LOOP
```

Se esegui questo programma il computer mostrerà ripetutamente la parola CIAO fino a quando non lo fermi premendo il tasto RUN/STOP. Questo succede perché stiamo dicendo al computer di far apparire la parola CIAO e poi di tornare al comando immediatamente dopo l'istruzione DO (che è di nuovo l'istruzione PRINT) e partire nuovamente.

Ciò può sembrare inutile. Comunque, se fai questi cambiamenti nel programma incomincerai a pensarla diversamente.

```
10 DO UNTIL A=10
15 A=A+1
```

Quando questa volta farai eseguire il programma vedrai che il computer mostrerà la parola CIAO 10 volte e poi si fermerà. Questo succede perché stiamo dicendo al computer di fare ogni cosa tra l'istruzione DO e LOOP fino a quando (UNTIL) la variabile 'A' raggiunge il valore di 10. Se il valore di A parte da zero e viene aumentato di uno ogni volta che il computer torna indietro, la parola CIAO apparirà 10 volte. Se cambi la linea 10 con:

```
10 DO UNTIL A=20
```

allora la parola CIAO apparirà 20 volte.

Un'altra forma del DO ... LOOP è il DO WHILE ... LOOP. È molto simile al ciclo DO UNTIL ... LOOP, come vedrai se cambi la linea 10 con:

```
10 DO WHILE A<10
```

Quando fai eseguire il programma vedrai apparire la parola CIAO dieci volte, esattamente come prima. Questa volta il computer eseguirà ogni cosa tra le istruzioni DO e LOOP solo mentre (WHILE) il valore della variabile è minore di 10. Smetterà solo quando A raggiunge il valore di 10.

Le istruzioni WHILE e UNTIL possono essere inserite dopo il DO o dopo il LOOP, in modo da poter cambiare la linea 10 e la linea 30 in questo modo:

```
10 DO
30 LOOP WHILE A<10
```

e il programma lavorerà nello stesso modo.

Qualche volta è necessario lasciare il ciclo DO ... LOOP a metà strada; per farlo usiamo l'istruzione EXIT (USCITA). Prova aggiungendo queste linee al tuo programma:

```
25 GET A$:IF A$="A" THEN EXIT
```

Ora puoi fermare il programma premendo il tasto A. Ciò succede perché è stato detto al computer di esaminare la tastiera e se il tasto A è stato premuto, allora deve uscire (EXIT) dal ciclo.

È possibile nidificare i cicli DO ... LOOP, cioè infilarli uno nell'altro, proprio come si faceva con i cicli FOR ... NEXT. Ecco un esempio:

```
10 SCNCLR
20 DO UNTIL Y=25
30 DO UNTIL X=40
40 CHAR 1,X,Y,"*"
50 FOR N=1 TO 50:NEXT N
60 CHAR 1,X,Y," "
70 X=X+1:LOOP
80 X=0:Y=Y+1:LOOP
```

Il programma funziona così:

Linea 10: Pulisce lo schermo.
Linea 20: Esegue tutto ciò che c'è tra le istruzioni DO e LOOP fino a quando il valore della variabile Y raggiunge 25.
Linea 30: Esegue tutto ciò che c'è tra le istruzioni DO e LOOP fino a quando il valore della variabile X raggiunge 40.
Linea 40: Appare una stella di coordinate X,Y.
Linea 50: Ciclo di ritardo.
Linea 60: Appare uno spazio alle stesse coordinate X,Y della stella.
Linea 70: Aggiunge uno al valore della variabile X poi si ricollega all'istruzione dopo l'ultimo DO.
Linea 80: Assegna il valore zero alla variabile X, aggiunge uno al valore della variabile Y e si ricollega alla prima istruzione DO.

Il ciclo centrale viene eseguito prima del ciclo più esterno e viene eseguito 40 volte per ogni incremento del ciclo esterno. Il ciclo centrale controlla la posizione orizzontale della stella e poi eseguendo questo ciclo 40 volte fa muovere la stella verso la destra dello schermo un carattere alla volta. La stella si muove in orizzontale di una linea ogni volta che il ciclo centrale ha completato i 40 giri fino a raggiungere il fondo dello schermo.

Riassunto

Tutte le istruzioni tra il DO e il LOOP vengono eseguite finché (UNTIL) una condizione viene realizzata o mentre (WHILE) viene realizzata.

Le istruzioni UNTIL e WHILE possono venire dopo il DO o dopo il LOOP.

L'istruzione EXIT farà fermare il computer mentre esegue il ciclo, sia che questo sia finito o no.

I cicli DO ... LOOP possono essere nidificati esattamente come succede per i cicli FOR ... NEXT.

DIM e matrici

La matrice è una forma di variabile un po' più complessa rispetto a quelle usate fino ad ora. Essa è più utile perché ha un indice.

Queste matrici saranno difficili da capire all'inizio. Immagina un libro con 12 pagine, numerate da 0 a 11; su ogni pagina è scritto un numero. Questo libro rappresenta una variabile numerica dimensionata, detta anche matrice. Se per es. vuoi vedere cosa è scritto a pag. 5 del libro, lo prenderai, lo sfoglierai fino a pag. 5 e leggerai cosa è scritto.

Una matrice è simile al libro con le sue pagine numerate. Funziona in modo simile; se per es. hai una variabile dimensionata chiamata LIBRO, e vuoi vedere quale era il quinto numero rappresentato da LIBRO, digiterai:

```
PRINT LIBRO (5)
```

Il computer cercherà allora la variabile dimensionata LIBRO, vedrà da che cosa è rappresentato il quinto numero e lo mostrerà sullo schermo.

Sfortunatamente queste variabili utilizzano una grande quantità della memoria del computer, così è importante dire al computer quante variabili dimensionate vuoi usare e quanto vuoi immagazzinare in esse. Per farlo devi usare il comando DIM (DIMENSIONARE). Per es. se vuoi immagazzinare 14 numeri in una matrice chiamata LIBRO, allora userai una linea come questa:

```
10 DIM LIBRO (13)
```

Così dici al computer di riservare abbastanza memoria per 14 numeri (da 0 a 13 sono 14 numeri). Puoi aggiungerlo al programma in modo che esso chieda 14 numeri e poi li stampi:

```
20 SCNCLR
30 FOR N=0 TO 13
40 INPUT"PER FAVORE DIGITA UN NUMERO";LIBRO(N)
50 NEXT N
60 SCNCLR
70 FOR N=0 TO 13
80 PRINT LIBRO(N)
90 NEXT N
```

Quando fai eseguire questo programma il computer ti chiederà di digitare 14 numeri e poi lo schermo si pulisce e tutti i numeri che hai digitato verranno mostrati sullo schermo. Ecco una facile versione del programma:

Linea 10: Tiene da parte abbastanza memoria per immagazzinare 14 numeri nella variabile dimensionata (matrice) LIBRO.
Linea 20: Pulisce lo schermo.
Linea 30: Parte ripetendo tutti i comandi tra FOR e NEXT 14 volte.
Linea 40: Appare il messaggio PER FAVORE DIGITA UN NUMERO, aspetta perché tu inserisca il numero e poi assegna quel numero alla parte N della matrice.
Linea 50: Segna la fine del ciclo FOR ... NEXT.
Linea 60: Pulisce lo schermo.
Linea 70: Parte ripetendo tutti i comandi tra FOR e NEXT 14 volte.
Linea 80: Appare il numero rappresentato dalla parte N della matrice LIBRO.
Linea 90: Segna la fine del ciclo FOR ... NEXT.

Il programma sopracitato ha usato una variabile ad una dimensione e l'ha paragonata con il libro. Comunque possiamo usare anche due variabili dimensionate. Paragonandole al nostro libro possiamo, non solo scegliere quale pagina guardare, ma anche quale libro tra una serie di libri su uno scaffale. La variabile ha due numeri indice, così possiamo dire a quale libro e a quale pagina facciamo riferimento.

Due variabili dimensionate sono utili per le tabelle. Per es. se vogliamo mettere i risultati di una serie di gare in uno schema, potrebbero apparire così:

	Primo	*Secondo*	*Terzo*
100 m maschile	Paolo	Marco	Raimondo
400 m maschile	Giorgio	Martino	Guglielmo
100 m femminile	Sara	Giovanna	Carolina
400 m femminile	Maria	Clara	Giulia

Questo tipo di schema può essere rappresentato con una variabile bidimensionale, abbastanza facilmente. Per farlo abbiamo bisogno di una variabile bidimensionale che può contenere 12 elementi (ogni numero o gruppo di caratteri assegnati ad una variabile dimensionata viene chiamato elemento) 3 elementi in orizzontale e 4 in verticale. La variabile dimensionata dovrà essere naturalmente una variabile stringa perché dobbiamo immagazzinare dei nomi. Questo illustra l'argomento abbastanza bene:

```
 10 SCNCLR
 20 DIM PIAZZ$(2,3)
 30 PIAZZ$(0,0)="PAOLO":PIAZZ$(1,0)="MARCO":PIAZZ$(2,0)="RAIMONDO"
 40 PIAZZ$(0,1)="GIORGIO":PIAZZ$(1,1)="MARTINO":PIAZZ$(2,1)="GUGLIELMO"
 50 PIAZZ$(0,2)="SARA":PIAZZ$(1,2)="GIOVANNA":PIAZZ$(2,2)="CAROLINA"
 60 PIAZZ$(0,3)="MARIA":PIAZZ$(1,3)="CLARA":PIAZZ$(2,3)="GIULIA"
 70 PRINT TAB(9);"RISULTATI GARA":PRINT
 80 PRINT"1) 100M (MASCH)":PRINT"2) 400M (MASCH)":PRINT"3) 100M (FEMM)"
 85 PRINT"4) 400M (FEMM)"
 90 PRINT:INPUT"DI QUALE GARA VUOI I RISULTATI";GARA
100 IF GARA>4 OR GARA<1 THEN GOTO 90
110 INPUT"QUALE POSIZIONE VUOI CONOSCERE";PS
120 IF PS>3 OR PS<1 THEN GOTO 110
130 GARA=GARA-1:PS=PS-1
140 PRINT"LA PERSONA CHE HA CONQUISTATO QUEL POSTO IN QUELLA GARA ERA"
150 PRINT PIAZZ$(PS,GARA)
160 PRINT:PRINT"PREMI UN TASTO QUALSIASI PER AVERE UN ALTRO RISULTATO"
170 GETKEY A$:SCNCLR:GOTO 70
READY.
```

Il programma funziona così:

Linea 10: Pulisce lo schermo.
Linea 20: Tiene da parte abbastanza memoria per una variabile stringa dal nome PIAZZ$ con spazio sufficiente per 3 elementi di 4 elementi ciascuno.
Linea 30-60: Assegna i nomi ad ogni elemento della variabile stringa PIAZZ$; ad es. assegna il nome PAOLO all'elemento (0,0) della variabile PIAZZ$, assegna il nome MARCO all'elemento (1,0) della variabile PIAZZ$.
Linea 70: Appare il messaggio RISULTATI GARA sullo schermo con la prima lettera del messaggio nella nona posizione verticale e poi appare una linea vuota.
Linea 80-85: Appare il messaggio '1) 100M(MASCHILE)', '400M(FEMMINILE)' ecc.
Linea 90: Appare una linea vuota e il messaggio 'DI QUALE GARA VUOI I RISULTATI' e aspetta la risposta che dovrà essere assegnata alla variabile GARA.
Linea 100: Se il valore della variabile GARA è più grande di 4 o il valore della variabile GARA è minore di 1, allora va alla linea 90 e continua il programma da quel punto.
Linea 110: Appare il messaggio 'QUALE POSIZIONE VUOI CONOSCERE' e assegna la risposta alla variabile PS.

Linea 120: Se il valore dela variabile PS è più grande di 3 o minore di 1, allora va alla linea 110 e continua il programma da quel punto.
Linea 130: Sottrae uno dal valore della variabile GARA e assegna il risultato alla variabile GARA, poi sottrae uno dalla variabile PS e assegna il risultato alla variabile PS.
Linea 140: Appare il messaggio 'LA PERSONA CHE HA CONQUISTATO QUEL POSTO IN QUELLA GARA ERA'.
Linea 150: Trova il valore dell'elemento (PS,GARA) della variabile PIAZZ$ e lo mostra sullo schermo.
Linea 160: Appare una linea vuota e poi il messaggio 'PREMI UN TASTO QUALSIASI PER AVERE UN ALTRO RISULTATO'.
Linea 170: Aspetta che tu prema un tasto, poi pulisce lo schermo e va alla linea 70 e continua il programma da quel punto.

L'ampiezza della tua variabile dimensionata è ristretta solo alla quantità di memoria disponibile, così potrai avere una variabile come A (4,4,4,4,4,4) se hai abbastanza spazio in memoria. Comunque è improbabile dover usare delle variabili dimensionate più grandi di una variabile tridimensionale ad es. A(5,5,5).

Riassunto

L'istruzione DIM è usata per riservare dello spazio in memoria per una variabile dimensionata (matrice).

Un elemento è un numero o una stringa di caratteri che sono stati assegnati ad una parte della variabile.

Una variabile dimensionata è costituita da un nome di variabile con una serie di numeri di indice. Essi devono essere tra parentesi e dicono al computer a quale parte della variabile si fa riferimento.

L'ampiezza della variabile dimensionata è limitata solo alla quantità di memoria che hai a disposizione.

CAPITOLO 6

METTIAMO UN PO' D'ORDINE

CHR$

Ti sarai domandato come fa esattamente il computer ad immagazzinare caratteri in memoria. È ovvio che esso non può realmente immagazzinare la forma di ogni carattere di un programma nella sua memoria, così dà un codice ad ogni carattere. Questo codice è detto codice ASCII. Per es. la lettera A ha il codice 65, il cuore ha il codice 115 e così via.

Il comando CHR$ ti permette di accedere a questi codici e farli apparire sullo schermo. Se digiti:

```
PRINT CHR$(65)
```

vedrai apparire sullo schermo la lettera A, perché essa ha il codice 65. Similmente, se digiti:

```
PRINT CHR$(115)
```

vedrai apparire un cuore. Puoi provare a fare degli esperimenti con i codici di caratteri differenti e vedere quale effetto hanno (scegli i codici dello schermo alla Appendice C, ma non usare un numero più grande di 255 perché non ci sono più caratteri dopo quello con il codice 255).

Alcuni codici CHR$ sono codici di controllo. Sono per la maggior parte dallo 0 al 31, sebbene ce ne siano anche degli altri. I codici di controllo sono dei caratteri speciali che fanno muovere il cursore intorno allo schermo, cambiano il colore del testo e così via.

TAB

Il comando TAB è usato sempre con il comando PRINT e serve per decidere in quale colonna (o quante posizioni carattere in orizzontale) vuoi che inizi la stampa. Per es.:

```
PRINT TAB(30);"CIAO"
```

farà apparire il messaggio CIAO con la C nella posizione 30 orizzontale, la I nella posizione 31 e così via.

Una cosa importante da ricordare è che, se per es., digiti questo comando:

```
PRINT TAB(30);"CIAO";TAB(10);"LA'"
```

dove la seconda parola che deve apparire è dietro alla prima, allora la seconda verrà stampata sulla linea verticale seguente rispetto alla prima, anche se partirà dalla posizione 10 orizzontale.

DELETE (CANCELLARE)

Immagina di scrivere un lungo programma e di decidere che non hai bisogno di una certa ROUTINE che è lunga circa 15 linee. Impiegheresti del tempo per cancellarle tutte digitando il numero di linea di ognuna e poi premendo il RETURN. Per risparmiarti questa digitazione in più il tuo computer è provvisto di un comando che cancella parecchie linee di programma per volta. È il comando DELETE.

Esso è usato in modo simile al comando LIST, tranne che per lo sviluppo delle linee del programma; infatti il DELETE le toglie. Per es. per cancellare dalla linea 90 alla linea 150 compresa devi digitare:

```
DELETE 90-150
```

Per cancellare tutte le linee fino alla numero 75 compresa digiterai:

```
DELETE -75
```

E per cancellare tutte le linee dalla 5010 inclusa in avanti dovrai digitare:

```
DELETE 5010-
```

RENUMBER (RINUMERARE)

Il comando RENUMBER è molto utile quando stai scrivendo un tuo programma, perché ti permette di rimettere in ordine i numeri di linea. Immagina per es. che stai scrivendo un tuo programma e hai numerato le linee di 10 in 10. Poi scopri che hai bisogno di aggiungere 13 linee tra la n. 50 e la n. 60. Non c'è ovviamente nessuna soluzione per fare in modo che ti stiano, e allora cosa fai? Rinumeri le linee di 20 in 20 e hai spazio a sufficienza per aggiungere linee in più, così come potrai fare con tutte quelle che avrai bisogno di aggiungere più avanti.

Puoi rinumerare un programma seguendo uno dei parecchi modi possibili. Essi sono:

`RENUMBER` Rinumera l'intero programma con la prima linea che diventa la n. 10 e le seguenti incrementate di 10.

`RENUMBER 50,,7` Rinumera il programma con la linea 7 che diventa la n. 50 e le seguenti con un incremento di 10.

`RENUMBER 100,20,15` Rinumera il programma con la linea 15 che diventa 100 e le seguenti con un incremento di 20.

`RENUMBER 200,40` Rinumera il programma con la prima linea che diventa 200 e le seguenti con un incremento di 40.

REM

Quando scrivi un programma da solo, specialmente se è abbastanza lungo con parecchie sottoprocedure, è spesso utile essere capace di mettere qua e là dei comandi che ti aiutino a ricordare cosa fa ogni parte. Il comando REM (che è la abbreviazione di REMark = commento) ti permette di fare tali commenti che però vengono ignorati dal computer. Per es. questa linea ti aiuterà a ricordare che la procedura seguente muove un pezzo a sinistra:

```
530 REM MUOVE UN PEZZO A SINISTRA
```

Puoi mettere tutto quello che vuoi dopo un REM e il computer lo ignorerà.

END (FINE)

Abbiamo già usato il comando END un paio di volte nei nostri programmi, così dovresti già sapere che esso dice al computer di fermarsi nell'esecuzione del programma, anche se non ha ancora raggiunto l'ultima linea. Sembrerebbe inutile, ma avrai bisogno di fermare un programma prima dell'ultima linea se ci sono delle sottoprocedure alla fine del programma.

STOP (FERMARE)

Il comando STOP è simile al comando END eccettuato il fatto che è possibile far ricominciare il programma dopo averlo digitato, sapendo anche a che linea il programma si era fermato.

CONT (CONTINUARE)

Il comando CONT è usato per continuare un programma dopo che è stato fermato o dopo che hai premuto il tasto RUN/STOP. Questo comando farà continuare il programma dal punto in cui era stato lasciato e non azzererà nessuna variabile. In qualche caso però non è possibile continuare un programma, per es. se hai alterato una linea di programma dopo aver fermato il programma stesso. In questo caso ti verrà dato un CAN'T CONTINUE ERROR (NON POSSO CONTINUARE).

Finestre video

Le finestre video ti permettono di lavorare in un'area dello schermo in modo che il resto rimanga indisturbato. Tutto ciò che normalmente ha luogo sullo schermo in tutta la sua ampiezza, può aver luogo solo nella finestra. Per es. pulisci lo schermo (premendo SHIFT o CLEAR/HOME) poi premi il tasto con la freccia verso il basso cinque volte. Premi il tasto ESC, poi la T. Premi il tasto con la freccia verso il basso altre otto volte — poi premi quello con la freccia verso destra dieci volte. Premi ancora ESC e poi la B. Alla fine premi il tasto HOME, il cursore salterà nell'angolo in alto a sinistra, nella finestra.

Puoi digitare programmi, come puoi anche listarli o farli eseguire all'interno di una finestra e tutto ciò che digiti apparirà in questa finestra, come osserverai. La finestra funziona effettivamente come una versione ridotta dello schermo. Per uscire dalla finestra devi premere il tasto HOME due volte.

Premendo ESC e T contrassegni l'angolo in alto a sinistra della finestra, mentre premendo il tasto ESC e poi la B contrassegni l'angolo in basso a destra della finestra. Osserverai che non puoi muovere il cursore in su o verso sinistra dopo aver contrassegnato l'angolo in alto a sinistra della finestra e non puoi muoverlo fuori dalla finestra in qualunque direzione, una volta che la finestra è stata definita, fino a quando non premi il tasto HOME due volte.

Le finestre possono essere costruite abbastanza facilmente all'interno di un programma. Per farlo basta conoscere il codice del tasto ESC che è 27. Una volta che lo conosci, puoi costruire una finestra dove vuoi, tenendo conto che [5X CD] significa premere il tasto con la freccia verso il basso 5 volte e [4X CR] significa premere il tasto con la freccia verso destra 4 volte.

```
10 SCNCLR
20 PRINT"[5*CD][4*CR]";CHR$(27);"T"
30 PRINT"[10*CD][17*CR]";CHR$(27);"B[HOME]"
40 FOR N=1 TO 100:PRINT"UNA FINESTRA !";:NEXT
```

Come potrai vedere, costruire una finestra con un programma è simile a costruire una finestra in modo diretto. La misura e la posizione della finestra vengono definite muovendo il cursore nell'angolo in alto a sinistra della finestra e poi verso l'angolo in basso a destra. L'unica differenza è che i controlli del cursore sono tra virgolette e ogni volta che noi volessimo premere il tasto ESC mettiamo invece il comando CHR$(27) e dove normalmente digitiamo T o B, mettiamo le stesse lettere tra virgolette.

Le finestre sono utili se vuoi convertire un programma in BASIC di un computer diverso, per il tuo computer. Se la misura del testo sullo schermo è meno di 25 righe di 40 caratteri (colonne) allora puoi costruire una finestra della stessa ampiezza dello schermo dell'altro computer. Tutti i caratteri rimarranno dentro la finestra, per evitarti di modificare il programma.

Il tasto ESC

Abbiamo già usato il tasto ESC per definire la grandezza di una finestra, ma questo è un tasto molto versatile e ha anche altri usi. Quando è usato con uno dei tasti lettera normali, attiva delle funzioni speciali. Troverai un elenco di queste funzioni qui di seguito insieme ad una breve descrizione di cosa fa ognuna di esse:

ESC A	Attiva il 'modo inserimento';
ESC B	Definisce l'angolo in basso a destra della finestra;
ESC C	Disattiva il 'modo inserimento';
ESC D	Cancella la linea su cui si trova il cursore;
ESC I	Inserisce una linea;
ESC J	Muove il cursore all'inizio della linea corrente;
ESC K	Muove il cursore alla fine della linea corrente;
ESC L	Attiva lo scroll dello schermo;
ESC M	Disattiva lo scroll dello schermo;
ESC N	Fa ritornare lo schermo a grandezza normale;
ESC O	Disinserisce il lampeggiamento, la stampa in negativo, l'inserimento e il modo tra virgolette;
ESC P	Cancella tutto ciò che c'è sulla linea prima del cursore;
ESC R	Riduce lo sviluppo dello schermo a 38 × 23;
ESC T	Inserisce l'angolo in alto a sinistra della finestra;
ESC V	Fa scorrere lo schermo in su di una linea;
ESC W	Fa scorrere lo schermo in giù di una linea;
ESC X	Cancella la funzione ESC.

Premendo ESC e A (devi rilasciare il tasto ESC prima di premerne un altro) inserisci il 'modo di inserimento'. Così ogni cosa che digiterai sarà inserita tra ciò che è sulla linea corrente. Per es. digita:

```
10 PRINT "CIAO"
```

poi muovi il cursore verso il secondo gruppo di virgolette e digita ESC e poi A. Qualunque cosa digiti ora verrà inserita prima delle virgolette e si muoverà verso destra di uno spazio ogni volta che digiterai un altro carattere. Quando hai finito di inserire del testo nella linea, premi ESC e C e il modo di inserimento viene disattivato.

Potresti anche cancellare una linea intera e fare in modo che tutto ciò che c'è sotto quella linea, passi in su per riempire lo spazio vuoto. Per es. se fai eseguire il sopracitato programma di una linea per un momento e poi lo fermi, muovendo il cursore verso una qualsiasi posizione sullo schermo e digita ESC e poi D, vedrai che il contenuto della linea su cui è posizionato il cursore verrà cancellato e tutto ciò che era sullo schermo, sotto quella linea, passerà sopra.

L'opposto dell'azione di cancellare una linea è di inserirla. Premendo ESC e I farai in modo che quello che è sullo schermo passi più in giù di una linea, lasciandoti una riga vuota per scrivere.

Può essere utile talvolta muovere il cursore all'inizio della linea, quando è circa a metà di essa. Il metodo più veloce e più facile è di digitare ESC e J. Per es. se muovi il cursore in un punto sullo schermo, verso metà e poi premi ESC e J, il cursore salterà automaticamente all'inizio della linea su cui si trova.

Puoi anche muovere il cursore alla fine della linea corrente in modo simile, premendo ESC e poi K, esattamente nello stesso modo utilizzato per farlo ritornare all'inizio della linea.

Una cosa che può rivelarsi molto utile è l'abilità di disinserire lo scroll dello schermo. Se digiti ESC e poi M e muovi il cursore fino in fondo allo schermo, lo vedrai riapparire in cima. Questo sistema è conosciuto come 'wraparound' cioè girare intorno. Se ora vuoi listare un programma, vedrai che quando lo schermo diventa completo non gira in su per permettere che il resto del programma possa apparire, invece il listato va fino in fondo allo schermo e passa poi all'inizio. Tutto può tornare alla situazione normale premendo ESC e L. La funzione ESC O cancellerà il modo di lampeggio, il negativo, le virgolette e l'inserimento, se essi sono stati inseriti.

Per es. digita CONTROL e FLASH ON e digita poche lettere. Ora premi ESC e poi O, digita ancora poche lettere e vedrai che esse non lampeggiano più. La stessa cosa succede se sono inseriti i modi in negativo, di inserimento e virgolette (il modo tra virgolette si ha quando hai aperto delle virgolette e ogni volta che premi un tasto controllo cursore appare invece un simbolo).

Quando editi un programma può essere utile poter cancellare delle intere parti di linea. Le funzioni ESC P e ESC Q ti permettono di farlo. Se digiti ESC P allora ogni cosa prima del cursore verrà cancellata con entrambe queste funzioni.

L'immagine che il computer fa apparire sulla televisione, qualche volta è troppo larga per adattarsi allo schermo. Il miglior modo per risolvere questo problema è di usare una televisione diversa, ma se non ne hai un'altra a tua disposizione, puoi ridurre la misura dello sviluppo dello schermo, digitando ESC e R. Lo schermo diventerà chiaro, lasciando un bordo tutto intorno allo schermo della

larghezza di un carattere. Potresti usare se vuoi, il tasto ESC per far girare lo schermo in su o in giù. Premendo ESC e V lo schermo scorrerà in su, mentre premendo ESC e W lo schermo scorrerà in giù.

Se premi per sbaglio il tasto ESC, devi poi premere il tasto X che annulla la funzione ESC.

Le funzioni ESC sono utilizzate al meglio quando sono inserite in un programma. Hai già usato in questo modo le funzioni ESC per definire la misura di una finestra dal BASIC, così devi sapere che per implementare queste funzioni devi inserire PRINT CHR$(27) e poi inserire la lettera che avresti digitato normalmente tra virgolette, così:

```
PRINT CHR$(27);"W"
```

farà passare in giù di una linea lo schermo, come succede quando hai premuto ESC e poi W.

Queste funzioni possono essere estremamente utili nei programmi e nel modo di comando. Sarebbe una buona idea assicurarsi di conoscere come usarle in modo appropriato, perché sarai sorpreso di constatare quanto spesso esse siano utili.

CAPITOLO 7

TUTTI FACCIAMO DEGLI ERRORI

Intercettazione degli errori (ERROR Trapping)

Un programma per computer per essere realmente buono deve essere facile da usare; ciò significa che per poco che sappia chi usa il programma e qualunque cosa faccia, anche stupida, il programma sia in grado di continuare a lavorare e dire all'utente che cosa ha sbagliato. Per aiutarti a scrivere tali programmi il tuo computer è stato provvisto di una facilitazione per l'intercettazione degli errori, così che, per es., se premi il tasto RUN/STOP quando non vorresti, il programma sarà in grado di spiegare cosa hai fatto di sbagliato.

Ecco un breve programma che illustra il concetto.

```
10 TRAP 90
20 SCNCLR
30 PRINT"PER FAVORE NON PREMERE IL TASTO RUN/
   STOP"
40 PRINT"MENTRE RIEMPIO LO SCHERMO DI O"
50 FOR N=1 TO 500:NEXT N
60 FOR N=1 TO 1024:PRINT"O";:NEXT N
70 FOR N=1 TO 500:NEXT N
80 RUN
90 SCNCLR
100 IF ER=30 THEN PRINT"TI HO DETTO DI NON PR
    EMERE QUEL TASTO!":ELSE STOP
110 PRINT"ORA RICOMINCIAMO"
120 PRINT"MA NON PREMERE IL TASTO RUN/STOP"
130 FOR N=1 TO 2000:NEXT N
140 RESUME 20
```

La linea 10 contiene il primo comando di ERROR trapping — TRAP. Questo comando dice al computer verso quale linea dirigersi quando si imbatte in un errore. In questo caso il comando TRAP dice al computer che se c'è un errore deve andare alla linea 90 e continuare con il programma da quel punto.

Il resto del programma è diretto fino a quando non arrivi alla linea 100. In questa linea si fa riferimento ad una variabile ER alla quale non è stato assegnato nessun valore dal computer. Questa è una variabile di sistema cioè una variabile usata dal computer. ER contiene il codice dell'ultimo errore che è stato fatto. Se il numero dell'errore per aver premuto il tasto RUN/STOP (premere il tasto RUN/STOP viene classificato dal computer come un errore) è 30, il computer eseguirà tutte le istruzioni dalla linea in avanti solo se il tasto RUN/STOP è stato premuto.

La linea 140 ha un altro comando. Il RESUME permette al computer di continuare con il programma principale dopo che è stato fatto un errore. Il RESUME è simile al GOTO, tranne per il fatto che contrassegna la fine della procedura di ERROR trapping, così quando il computer arriva alla linea 140 si rende conto che questa è la fine della sottoprocedura ERROR trapping e allora salta alla linea 20 e continua con il programma da quel punto.

C'è anche un'altra versione dell'istruzione RESUME — RESUME NEXT. Questa istruzione dice al computer di andare verso il programma principale e di continuare con esso dalla prossima (NEXT) istruzione, dopo quella che ha causato l'errore. Per es., cambia la linea 140 con:

```
140 RESUME NEXT
```

poi digita RUN. Ad un certo punto premi il tasto RUN/STOP. Riceverai lo stesso messaggio che ti dice che non avresti dovuto premere quel tasto, e allora, dopo una breve pausa, il computer continuerà con il programma da dove si era fermato.

C'è anche un'altra variabile di sistema che è usata per gli errori. Questa variabile è EL e contiene il numero della linea in cui è stato commesso l'ultimo errore. Così se cambi la linea 10 con:

```
10 CIAO
```

e fai eseguire il programma, avrai subito un errore, naturalmente. Se ora digiti:

```
PRINT EL
```

il computer mostrerà il numero 10 che è il numero della linea in cui è avvenuto l'errore.

Sappiamo che ER contiene il numero dell'errore che è stato fatto per ultimo, ma un semplice numero non ti dice molto su quello che è successo. Per es. se ti dice che l'errore che hai appena fatto è il numero 11, non ne sai molto più di prima. Fortunatamente c'è una funzione che ti aiuta in questo caso — ERR$. Se digiti:

```
PRINT ERR$(11)
```

il computer mostrerà il messaggio SYNTAX sullo schermo. Ciò avviene perché l'errore numero 11 è un errore di sintassi. Analogamente se digiti:

```
PRINT ERR$(14)
```

vedrai apparire il messaggio ILLEGAL QUANTITY sullo schermo. Puoi usare la funzione ERR$ per trovare il messaggio per qualsiasi numero di errore, eccetto per quelli che si riferiscono ai disk drives (alle unità dischi).

Riassunto

È possibile intercettare un errore in modo che ogni volta che il computer ne incontra uno passerà subito alla tua procedura antierrore.

Il comando RESUME contrassegna la fine della procedura per il trattamento degli errori e dice anche al computer a quale linea saltare per continuare con il resto del programma.

Il sistema di variabili ER contiene il numero dell'ultimo errore che è stato fatto.

Il numero di linea in cui è stato fatto l'ultimo errore è indicato nella variabile di sistema EL.

Per trovare il messaggio d'errore che corrisponde ad ogni numero di errore, devi usare la funzione ERR$.

HELP (AIUTO)

Il comando HELP è estremamente utile quando stai tentando di trovare un errore in una linea di programma. Se per es. hai una linea con 4 o 5 comandi, e sai che c'è un errore in quella linea ma non sai dove, devi semplicemente digitare HELP e la linea con l'errore apparirà sullo schermo. Il comando con l'errore lampeggierà in modo che tu possa facilmente identificarlo. Il comando HELP funzionerà solo dopo che hai ricevuto un messaggio d'errore (premere il tasto contrassegnato con HELP ha lo stesso effetto che digitare HELP).

TRON E TROFF (TRACE ON e TRACE OFF)

È molto raro che un programma di una certa lunghezza e complessità funzioni subito la prima volta. Trovare errori reali (quelli che il computer riconosce come errori e te li comunica) non è un problema specialmente con il comando HELP. Comunque ci sono spesso dei BUGS (errori detti BUCHI) nel programma che, sebbene il programma funzioni, gli impediscono di fare esattamente quello che dovrebbe fare.

Per rendere più facile l'eliminazione di tutti i difetti, il tuo computer ha due comandi — TRON e TROFF. Il comando TRON dice al computer di attivare la traccia (TRACE ON). Quando ciò avviene il computer mostrerà il numero di linea della linea che sta eseguendo al momento sullo schermo, così appena puoi vedere l'errore, puoi vedere anche quale linea è stata eseguita e che sarà anche la linea con l'errore.

Per es. se vogliamo il messaggio sulla linea 20 del programma qui di seguito per leggere SALSICCIA E PURÈ invece di CROSTINI E FORMAGGIO possiamo usare il comando TRON per trovare la linea che mostra il messaggio CROSTINI E FORMAGGIO (lo so che puoi trovarla, ma in un programma con circa 200 linee sarebbe un po' difficile riconoscerla, e questo è un esempio). Così digita questo programma:

```
10 SCNCLR
20 PRINT"CROSTINI E FORMAGGIO"
30 FOR N=1 TO 10
40 PRINT"QUESTA E' UNA PROVA"
50 NEXT N
```

Siccome viene eseguita ogni linea vedrai il numero di linea apparire tra parentesi quadre ([ ]). Vedrai il [20] apparire appena il messaggio CROSTINI E FORMAGGIO appare sullo schermo. La ricerca rimarrà inserita fino a quando non la disinserisci con il comando TROFF (TRACE OFF) (traccia disattivata).

CAPITOLO 8

PROGRAMMAZIONE AVANZATA

READ, DATA e RESTORE

Spesso è utile avere un elenco di numeri o di caratteri a cui il computer può fare riferimento e che può usare. Questi numeri o caratteri possono essere un elenco di nomi e indirizzi, tra cui vuoi che il computer ne cerchi uno specifico, ad es.. Un modo per farlo è di immagazzinare il nome di ogni persona e l'indirizzo in una variabile stringa e possibilmente in una variabile dimensionata e cercare tra ogni variabile quando vuoi trovare il nome di una persona. La via più semplice sarebbe di avere un elenco di dati (DATA) tra cui il computer può esaminare per cercare il nome desiderato. Per inserire un elenco usiamo l'istruzione DATA, in questo modo:

```
1000 DATA "LUIGI ROSSI",123456,"GIOVANNI BIANCHI",123642
```

Questo è solo un breve elenco: solo due nomi e due numeri di telefono, ma l'elenco può continuare, usando parecchie istruzioni DATA. Come puoi vedere nell'elenco può essere incluso un qualsiasi carattere e può essere inserito tra virgolette, anche se non è essenziale. Anche i numeri possono essere immagazzinati nella lista, e sia i numeri che i caratteri possono essere mischiati liberamente, come puoi vedere.

Non è di grande utilità avere un elenco del genere, se non puoi farci niente. Quello che dobbiamo fare è leggere (READ) ogni parte di dati nella variabile, i numeri in una variabile numerica e gli altri caratteri in una variabile stringa (se vuoi anche i numeri possono essere letti in una variabile stringa, ma non potrai fare delle operazioni con essi). Questo breve programma è una agenda telefonica che ti chiede il nome delle persone e poi ti dice il loro numero di telefono.

```
10 SCNCLR:INPUT"QUAL'E' IL NOME DELLA PERSONA
   ";NOME$
20 FOR N=1 TO 5:READ A$,X
30 IF A$=NOME$ THEN PRINT"IL SUO NUMERO DI TE
   LEFONO E'";X:GOTO 60
```

```
40 NEXT N
50 PRINT"MI DISPIACE,MA NON CONOSCO IL SUO NU
   MERO DI TELEFONO"
60 PRINT:PRINT"PREMI UN TASTO"
70 GETKEY A$:RUN
80 DATA LUIGI ROSSI,123456,GIOVANNI BIANCHI,1
   23462,FABIO CESTARI,126463
90 DATA GIUSEPPE GRILLO,126457,PIETRO GIUSSAN
   I,327859
```

Quando esegui il programma lo schermo si pulirà e ti verrà chiesto il nome della persona della quale vuoi il numero di telefono (scegline uno dall'elenco delle istruzioni DATA). Il computer allora cercherà nell'elenco di DATA e se troverà il nome della persona, ti dirà il suo numero di telefono (che nell'elenco, è immediatamente dopo il nome). Se il computer non trova il nome della persona, te lo dirà.

Il programma funziona così:

Linea 10: Pulisce lo schermo. Appare il messaggio 'QUAL È IL NOME DELLA PERSONA' e poi aspetta la risposta prima di assegnarla alla variabile stringa NOME$.
Linea 20: Inizia la ripetizione di tutto quello che è tra i comandi FOR e NEXT per 5 volte, con il valore della variabile N che parte da 1 e aumenta di uno ogni volta che il ciclo si chiude con NEXT fino a raggiungere 5. Legge (READ) il pezzo seguente dei DATA dall'elenco e lo assegna alla variabile stringa A$, poi legge il pezzo seguente dei DATA e lo assegna alla variabile X.
Linea 30: Verifica se la variabile stringa A$ è uguale alla variabile stringa NOME$. Se lo è allora appare il messaggio 'IL SUO NUMERO DI TELEFONO È' e poi mostra il valore della variabile X prima di saltare alla linea 60 e continuare il programma da quel punto.
Linea 40: Contrassegna la fine del ciclo FOR ... NEXT.
Linea 50: Appare il messaggio 'MI DISPIACE, MA NON CONOSCO IL SUO NUMERO DI TELEFONO'.
Linea 60: Appare una linea vuota e poi il messaggio 'PREMI UN TASTO.'
Linea 70: Aspetta che tu prema un tasto e assegna il simbolo su quel tasto alla variabile stringa A$ prima di incominciare di nuovo il programma.
Linea 80: Lista di DATA.
Linea 90: Lista di DATA.

La linea 20 del programma legge (READ) il successivo pezzo di DATA dall'elenco alla fine del programma (in realtà legge due dati, ma uno per volta). Ogni volta che una parte di DATA viene letto, il computer ricorda dov'è il prossimo pezzo

di DATA, in modo che la volta seguente, quando si imbatterà in una istruzione READ sa da dove prendere i DATA.

Molti dati possono essere letti con una istruzione READ, come puoi vedere dal programma precedente. Tutto quello che devi fare è dire al computer quali variabili vuoi che siano assegnate ai DATA, nell'ordine corretto, e separare ogni variabile con una virgola. Puoi leggere quante parti di DATA vuoi, fino a quando l'elenco delle variabili sarà più corto di una linea di un programma normale; se non succederà dovrai usare due linee e due istruzioni READ.

Ma cosa succede quando il computer raggiunge la fine dell'elenco dei DATA? Bene, una volta arrivato a quel punto non ha più niente da leggere, così ti dà un OUT OF DATA ERROR (fuori dai DATA) se vuoi cercare di leggere altri DATA. Per es. se cambi la linea 70 del precedente programma con:

```
70 GETKEY A$:GOTO 10
```

ed esegui il programma per poco, alla fine riceverai il messaggio OUT OF DATA ERROR. Ciò succede perché il comando RUN (che all'inizio è stato usato nella linea 70) riporta l'indice dei DATA (quello che usa il computer per ricordare dov'è nell'elenco dei DATA) all'inizio dell'elenco, mentre un comando GOTO non lo fa. Per risolvere questo problema usiamo il comando RESTORE che dice al computer di incominciare a leggere i DATA ancora dall'inizio dell'elenco. Se ora cambiamo la linea 70 con:

```
70 GETKEY A$:RESTORE:GOTO 10
```

allora il programma funzionerà perfettamente.

Potresti anche dire al computer di incominciare a prendere dei DATA da un certo numero di linea. Per es. altera la linea 70 con:

```
70 GETKEY A$:RESTORE 90:GOTO 10
```

L'istruzione RESTORE 90 dice al computer di stabilire l'indice dei DATA all'inizio dei DATA sulla linea 90. Vedrai che potrai trovare solo gli indirizzi di Giuseppe Grillo e Pietro Giussani.

Avrai pensato fino ad ora che quando il computer raggiunge una istruzione READ salta alla linea su cui ci sono i DATA e la esamina per trovare la parte seguente dei DATA. Questo non è il nostro caso, comunque, e per verificarlo digita TRON e fai girare il programma. Vedrai che il computer non salta mai alla linea 80 o 90 per esaminare l'elenco dei DATA. Ciò succede perché il computer sa esattamente dove è immagazzinata, nella memoria, la parte seguente dei DATA, così non deve occuparsi della linea e va direttamente in quella posizione.

Riassunto

L'istruzione DATA viene usata per immagazzinare un elenco di caratteri e di numeri. I caratteri possono essere inseriti tra virgolette, anche se non è essenziale.

L'istruzione READ è usata per leggere una parte dei DATA e assegnarla ad una variabile. Questa istruzione può essere usata per leggere subito parecchie parti dei DATA e assegnare ogni parte ad una variabile separata. In questo caso le variabili nell'elenco, che seguono l'istruzione READ devono essere separate da una virgola.

L'istruzione RESTORE dice al computer di tornare indietro all'inizio dell'elenco dei DATA e incominciare a leggere i DATA da quel punto.

Il computer non va realmente alla linea che contiene i DATA, quando gli viene chiesto di leggere una parte dei DATA; sa esattamente dove sono immagazzinati i DATA e non deve occuparsi dei numeri di linea.

Uso delle stringhe

Le variabili stringa sono estremamente utili e ci sono tanti modi in cui puoi dividerle e poi riunirle di nuovo per adattarle ai tuoi bisogni. Ecco i comandi disponibili.

LEFT$

Se ti ricordi il capitolo su IF ... THEN ... ELSE ti ritorneranno in mente tre linee simili a queste:

```
130 IF A$="SI" OR A$="S" THEN ...
```

Sarebbe molto più facile se potessimo verificare se la prima lettera della A$ è una S, perché allora potremmo accettare delle risposte quale S,SI o praticamente qualunque altra versione della parola SI partendo dalla lettera iniziale S. Per farlo usiamo il comando LEFT$, come questo:

```
130 IF LEFT$(A$,1)="S" THEN ...
```

Questa linea particolare verifica qual è il primo carattere della variabile A$. Se cambi la linea 30 con:

```
130 IF LEFT$(A$,2)="SI" THEN ...
```

allora il computer verificherà quali sono i primi due caratteri della variabile A$ (il 2 nel comando LEFT$ dice al computer che vuoi i primi due caratteri; cambiandolo con un 3 significherebbe che vuoi i primi tre caratteri).

Naturalmente non devi usare una variabile stringa con il comando LEFT$ (con qualsiasi altro comando per l'uso delle stringhe si ottiene lo stesso risultato) e puoi anche usare dei caratteri tra virgolette, come questo:

```
250 IF LEFT$("COMPUTER",4)="COMP" THEN ...
```

Ecco un breve programma esemplificativo:

```
10 SCNCLR:INPUT"TI PIACE USARE IL COMPUTER";C
   O$
20 IF LEFT$(CO$,1)="S" THEN PRINT"NE SONO FEL
   ICE !"
30 IF LEFT$(CO$,1)="N" THEN PRINT"OH,NON SONO
    DA BIASIMARE"
```

RIGHT$

Il comando RIGHT$ è molto simile al comando LEFT$, se si eccettua il fatto che il comando LEFT$ verifica quali sono i primi tre caratteri della stringa, mentre il comando RIGHT$ verifica quali sono gli ultimi caratteri della stringa. Prova questo esempio:

```
10 SCNCLR:INPUT"DIGITA QUALCOSA PER FAVORE";Z
   Z$
20 PRINT"I PRIMI DUE CARATTERI CHE HAI DIGITA
   TO SONO:";
30 PRINT LEFT$(ZZ$,2):PRINT"GLI ULTIMI DUE CA
   RATTERI CHE HAI DIGITATO SONO:";
40 PRINT RIGHT$(ZZ$,2)
```

Il 2 nel comando RIGHT$ dice al computer che vuoi gli ultimi due caratteri della stringa e può essere cambiato facilmente, proprio come avviene per il comando LEFT$.

MID$

Il comando MID$ è usato per cercare quali sono i caratteri centrali, piuttosto che quelli all'inizio o alla fine. Anziché dire solo quanti caratteri vuoi verificare come fai con i comandi LEFT$ e RIGHT$, devi dire anche da dove vuoi che parta. Per es. se hai una linea come questa:

```
310 A$=MID$(B$,4,3)
```

allora il computer assegnerà 3 caratteri dal centro della variabile stringa B$ alla variabile stringa A$, con il primo di questi caratteri che è il quarto dei caratteri della variabile stringa B$.

È anche possibile rimpiazzare parti della variabile stringa usando il comando MID$. Ecco un breve programma esemplificativo, che funziona così:

```
10 SCNCLR
20 A$="CIAO CIAO   A TUTTI"
30 PRINT A$
40 MID$(A$,6,6)="LASSU'"
50 PRINT A$
```

Se guardi la linea 40 vedrai che il comando MID$ è stato usato per rimpiazzare il secondo CIAO con LASSÙ. Ciò viene fatto dicendo semplicemente al computer da che parte della variabile stringa vuoi partire (il settimo carattere in questo caso) e quanti caratteri vuoi sostituire (in questo esempio vogliamo sostituire 5 caratteri) e poi dire al computer quali caratteri vuoi sostituire a quelli vecchi. Come puoi vedere i caratteri nuovi devono essere inseriti tra virgolette.

INSTR

Il comando INSTR è usato per scoprire se una stringa è contenuta in un'altra stringa. Prova questo breve programma:

```
10 SCNCLR:A$="SOPRA LA PANCA LA CAPRA CANTA,S
   OTTO LA PANCA LA CAPRA CREPA"
20 PRINT INSTR(A$,"PANCA")
```

Quando lo esegui apparirà sullo schermo il numero 10 perché la P delle lettere PANCA è la decima lettera della variabile stringa A$. Quello che hai appena chiesto al computer è di cercare nella variabile stringa A$ se le lettere PANCA sono contenute in essa. Se esse sono contenute nella A$ allora il computer ti dirà esattamente dove la prima lettera di PANCA appare nella variabile stringa A$.

Se guardi la stringa A$ vedrai che le lettere PANCA appaiono due volte. Il computer troverà solo la prima apparizione dei caratteri che stai cercando. Per fare in modo che il computer trovi le altre lettere PANCA dovrai cambiare la linea 20 con:

```
20 PRINT INSTR(A$,"PANCA",12)
```

Quando fai eseguire il programma, apparirà il numero 39 sullo schermo. Questa volta il computer ha incominciato la ricerca delle lettere PANCA a partire dal dodicesimo carattere della stringa A$. In altre parole il computer cercherà tra le lettere LA CAPRA CANTA SOTTO LA PANCA LA CAPRA CREPA la prima apparizione delle lettere PANCA e trova che la P di PANCA è il trentanovesimo carattere della variabile stringa A$.

Se il computer non può trovare i caratteri che tu gli hai detto di cercare nella stringa darà 0 (zero) come risultato.

LEN

Il comando LEN viene usato per trovare la lunghezza (LENGHT) della variabile stringa, oppure quanti caratteri essa contiene. Questo breve programma illustra l'uso del comando:

```
10 SCNCLR:A$="SUPERCALIFRAGILISTICHESPIRALIDO
   SO!"
20 PRINT"LA VARIABILE STRINGA 'A$' CONTIENE";
30 PRINT LEN(A$);"CARATTERI !"
```

Come puoi vedere la variabile stringa a cui vuoi fare riferimento deve essere chiusa tra parentesi, dopo il comando LEN.

Riassunto

Il comando LEFT$ è usato per ottenere i primi caratteri di una variabile stringa.

Il comando RIGHT$ è usato per ottenere gli ultimi caratteri della variabile stringa.

Il comando MID$ è usato per ottenere i caratteri dal centro della variabile stringa.

Il comando INSTR è usato per trovare se una stringa è contenuta in un'altra. Esso ti darà la posizione del primo carattere di una stringa che stai cercando all'interno della stringa esaminata. Se la stringa che stai cercando non è contenuta in un'altra stringa ti darà 0 (zero).

Il comando LEN è usato per trovare la lunghezza della variabile stringa che deve essere chiusa tra parentesi dopo il comando LEN.

SOUND E VOLume

È ora di sperimentare le capacità sonore del tuo computer. Esso ha tre voci differenti e tu puoi usarne anche due insieme. Ciò significa che puoi avere una voce per il tono e una per il ritmo. Due delle voci producono toni e l'altra produce un rumore bianco (è utile per esplosioni e spari di fucile).

Comunque prima di poter produrre qualsiasi suono, dobbiamo inserire il livello di volume. Per farlo usiamo il comando VOL insieme ad un numero da 0 a 8 (8 è il volume massimo, 0 il volume minimo oppure la posizione spento). È meglio inserire il volume al massimo, perciò digita:

```
VOL 8
```

Ora tutto ciò che dobbiamo fare è scegliere una voce e una nota e decidere per quanto tempo deve suonare. Per es. se vogliamo far suonare la nota C della voce 1 per 3 secondi, allora digiteremo il comando:

```
SOUND 1,810,180
```

Il numero 1 dice al computer che vogliamo usare la voce 1, il numero 810 dice al computer che vogliamo suonare la nota C nella terza ottava e il numero 180 che vogliamo che la nota duri per 3 secondi (la durata è in 160esimi di secondo — una durata di 60 fa durare la nota per 1 secondo).

Prova a cambiare l'1 del comando SOUND con un 2 e poi con un 3 per vedere come sono differenti le voci. Come ho già detto prima è possibile avere due voci che suonano nello stesso momento. Puoi far suonare sia la voce 1 e 2 o la voce 1 e 3. Se digiti questa linea sentirai una nota suonata al massimo del rumore bianco:

```
SOUND 1,810,360:SOUND 3,917,360
```

Il comando SOUND può essere usato per suonare delle note e per creare effetti sonori. Ecco alcuni programmi che lo fanno:

Doctor Foster

```
10 SCNCLR:VOL 8:PRINT"DOCTOR FOSTER"
20 FOR N=1 TO 36:READ NO,DU
30 SOUND 1,NO,DU
40 NEXT N
50 END:FOR N=1 TO 5000:NEXT N
60 END
70 DATA 739,60,739,30,834,60,834,30,810,30,83
   4,30,810,30,798,60,770,30
80 DATA 739,60,739,30,798,30,770,30,739,30,77
   0,90,770,60,770,30,739,30
```

```
90 DATA 739,30,739,30,834,30,810,30,798,30,81
   0,30,834,30,810,30,854,30
100 DATA 834,30,810,30,798,30,798,30,798,30,8
    80,60,770,30,881,90,881,60
```

Telephone

```
10 SCNCLR
20 VOL 8
30 FOR M=1 TO 10
40 FOR N=1 TO 10:SOUND 1,650,1:SOUND 1,700,1:
   NEXT N
50 FOR N=1 TO 50:NEXT N:FOR N=1 TO 10:SOUND 1
   ,650,1:SOUND 1,700,1:NEXT N
60 FOR N=1 TO 1000:NEXT N,M
70 FOR N=1 TO 8:SOUND 1,650,1:SOUND 1,700,1:N
   EXT N
80 SOUND 1,920,20
90 FOR N=8 TO 0 STEP -1:VOL N:FOR M=1 TO 5:NE
   XT M,N
```

Il computer può continuare ad eseguire altre istruzioni mentre sta eseguendo un suono, così può avere una nota che suona in sottofondo mentre sta succedendo qualcos'altro.

Riassunto

Il livello del volume viene selezionato dalla istruzione VOL. Esso può essere da 0 a 8.

Ci sono tre voci disponibili — le voci 1 e 2 o le voci 1 e 3 possono essere suonate contemporaneamente.

Una nota può essere di qualsiasi lunghezza da 0 a 65.535 sessantesimi di secondo.

Possono essere realizzati 1.024 suoni diversi. Fai riferimento alla seguente tabella per il numero di ogni nota musicale.

Note	Frequenza (Hz)	Valore
LA	110	11
SI	123.5	122
DO	130.8	173
RE	146.8	266

MI	164.7	349
FA	174.5	387
SOL	195.9	457
LA	220.2	520
SI	246.9	575
DO	261.4	600
RE	293.6	647
MI	330	689
FA	349.6	708
SOL	392.5	743
LA	440.4	774
SI	494.9	802
DO	522.7	814
RE	588.7	838
MI	658	858
FA	699	868
SOL	782.2	885
LA	880.7	901
SI	989.9	915
DO	1045	921
RE	1177	933
MI	1316	943
FA	1398	948
SOL	1575	957

Questa tabella contiene il valore del suono per 4 ottave ma i diesis e i bemolle non appaiono. La frequenza della nota è data solo come riferimento. Il valore indicato per ogni nota è quello che puoi usare come secondo numero dopo l'istruzione SOUND. Per es. per suonare la nota C per 1/2 secondo userai il comando:

```
SOUND 1,173,30
```

Puoi suonare una nota di qualsiasi frequenza, se conosci la frequenza della nota che desideri. Puoi comunque calcolare il valore da usare nella istruzione SOUND usando questa formula:

VALORE = 1024 – (110840,45/FREQUENZA).

ON ... GOTO e ON ... GOSUB

Nel capitolo sul GOTO e GOSUB è stato detto che non puoi usare una variabile al posto di un numero di linea con questi comandi. Per conciliare il tutto il tuo computer è stato provvisto dei comandi ON ... GOTO e ON ... GOSUB. Essi andranno ad una linea a seconda del valore della variabile. Per es. se il computer incontrasse questa linea:

```
120 ON ZZ GOTO 1000,2000,3000,4000
```

allora andrebbe a vedere quale numero rappresenta la variabile ZZ e poi andrà (GOTO) verso una delle linee seguenti a seconda del valore. Se il valore di ZZ è 1 allora il computer andrà alla linea 1000. Se il valore è 2 allora andrà alla linea 2000 e così via.

Il comando ON ... GOSUB funziona esattamente nello stesso modo del comando ON ... GOTO, eccettuato il fatto che esso va verso una sottoprocedura che dovrebbe terminare con l'istruzione RETURN, proprio come il comando GOSUB normale.

Ecco un breve programma che simula un dado e usa il comando ON ... GOSUB, per es.:

```
 10 SCNCLR
 20 FOR N=9 TO 13:CHAR 1,17,N,"[PURPLE][RVS ON
    ]     ":NEXT
 30 D=INT(RND(0)*6)+1:IF D=7 THEN GOTO 30
 40 ON D GOSUB 60,70,90,110,130,150
 50 FOR N=1 TO 1000:NEXT:RUN
 60 CHAR 1,19,11,"*":CHAR 1,17,14,"UNO":RETURN
 70 CHAR 1,18,10,"*":CHAR 1,20,12,"*":
 80 CHAR 1,17,14,"DUE":RETURN
 90 CHAR 1,18,10,"*":CHAR 1,19,11,"*":CHAR 1,2
    0,12,"*"
100 CHAR 1,17,14,"TRE":RETURN
110 CHAR 1,18,10,"* *":CHAR 1,18,12,"* *"
120 CHAR 1,17,14,"QUATTRO":RETURN
130 CHAR 1,18,10,"* *":CHAR 1,19,11,"*":CHAR
    1,18,12,"* *"
140 CHAR 1,17,14,"CINQUE":RETURN
150 CHAR 1,18,10,"* *":CHAR 1,18,11,"* *":CHA
    R 1,18,12,"* *"
160 CHAR 1,17,14,"SEI":RETURN
```

Le parentesi quadre nella linea 20 indicano che dovresti cambiare il colore del testo in color porpora e inserire la stampa in negativo premendo il CONTROL 5 e poi CONTROL 9.

Il programma funziona così:

Linea 10: Pulisce lo schermo.
Linea 20: Cambia il colore del testo in porpora e stampa un blocco solido di 5 linee di profondità e della larghezza di 5 caratteri; con il primo blocco nella posizione 17 orizzontale e di N caratteri in verticale.
Linea 30: Sceglie un numero a caso da 0 a 1, lo moltiplica per 6 e lo arrotonda prima di aggiungere 1 al risultato. Assegna il numero finale alla variabile D. Se il valore della variabile D è 7 allora esegue di nuovo questa linea (scegliendo un numero casuale da 1 a 6, poi aggiungendo 1 e arrotondandolo si otterrà un 7. Il computer sceglie molto raramente il massimo numero possibile quando sta scegliendo un numero a caso e comunque RND (0) 6 è sempre meno di 7).
Linea 40: Se il valore della variabile D è 1 vai (GOSUB) alla sottoprocedura che inizia alla linea 70. Se è 3 vai alla sottoprocedura che inizia alla linea 90 e così via.
Linea 50: Il ciclo FOR ... NEXT è vuoto e causa un ritardo prima di eseguire di nuovo il programma.
Linea 60: Appare una singola stella alla posizione 19 orizzontale e 11 verticale e poi appare il messaggio UNO con la O nella 17a colonna orizzontale e nella 14a riga verticale prima di ritornare al comando immediatamente dopo il comando GOSUB che ha chiamato questa sottoprocedura.
Linea 70: Appare una singola stella nella posizione 18 orizzontale e 10 verticale e poi appare un'altra stella nella posizione 20 orizzontale e 12 verticale.
Linea 80: Appare il messaggio DUE con la D sulla 17a colonna orizzontale e 14a riga verticale.

Tutte le linee dopo la nùmero 80 sono simili alla 60-80; appaiono solo diversi numeri di stelle.

AUTO

Se stai digitando un programma prendendolo da un libro o da una rivista, potrebbe diventare noioso digitare tutti i numeri di linea, specialmente se le linee sono numerate regolarmente (ad es. 10, 20, 30 e così via). Per velocizzare la digitazione di tali programmi ed anche per renderlo meno noioso, il tuo computer è stato provvisto di un comando di linea AUTOmatico. Se digiti:

```
AUTO 10
```

e poi parti digitando un breve programma, vedrai che appena premi il RETURN alla fine della prima linea, il numero di linea seguente apparirà automaticamente.

Il numero dopo il comando AUTO dice al computer qual è l'intervallo tra una linea e l'altra, così se digiti AUTO 50 il computer numererà le linee 50, 100, 150 e così via.

Una volta che arrivi alla fine del programma, devi premere il tasto RETURN senza digitare niente. Per es. se hai finito il tuo programma alla linea 1510 e il computer mostra il numero di linea seguente, cioè 1520 e aspetta che tu scriva qualcosa, dovrai solo premere il RETURN senza digitare niente altro.

CLEAR

In qualche circostanza avrai bisogno di azzerare tutte le variabili nel bel mezzo di un programma. Il modo più semplice per farlo è usare il comando CLR (CLEAR). Il programma non si ferma e non viene alterato in alcun modo.

Il comando CLR viene eseguito automaticamente quando alteri una linea di programma, o lo esegui.

ASC

Spesso è utile saper trovare il codice ASCII (quello di CHR$) per un carattere. Fortunatamente il tuo computer ha un comando che ti permette di trovare il codice CHR$ per qualsiasi carattere senza doverlo guardare sulla tabella. Questo comando è ASC.

Se digiti:

```
PRINT ASC("A")
```

vedrai apparire il numero 65 sullo schermo. 65 è il codice CHR$ per il simbolo A. In questo modo puoi trovare il codice CHR$ per qualsiasi carattere; tutto quello che devi fare è di inserire il carattere tra virgolette, come nell'es. precedente.

VAL

Il comando VAL è una funzione che ti dà il valore della variabile stringa. Per es. se alla variabile stringa BS$ è stato assegnato il 921 allora il comando:

```
Z=VAL(BS$)
```

assegnerà il numero 921 alla variabile Z.

Se c'è una combinazione di lettere e numeri in una variabile stringa si parte da sinistra. Se la stringa inizia con un numero allora il valore risultante dal comando VAL sarà il valore di quel numero (ad es. PRINT VAL ("A12") darà il valore 0).

STR$

Il comando STR$ è l'opposto di VAL perché questa funzione converte un numero in una stringa. Per es. se hai una linea così:

```
320 A$=STR$(864)
```

allora alla variabile stringa A$ verrà assegnato il carattere 864.

Il comando STR$ aggiungerà sempre uno spazio prima del numero all'inizio della variabile stringa in cui i caratteri sono stati immagazzinati. Per es. se digiti questo programma:

```
10 SCNCLR:A$=STR$(864)
20 PRINT A$:PRINT LEN(A$)
30 A$=RIGHT$(A$,3)
40 PRINT A$:PRINT LEN(A$)
```

vedrai che è stato aggiunto uno spazio prima del numero, quando è stato immagazzinato nella variabile stringa. La linea 30 toglie questo spazio e quando la stringa appare una seconda volta insieme al numero del carattere nella stringa, vedrai che lo spazio è stato tolto.

CAPITOLO 9

STAMPA E GRAFICI

PRINT USING

PRINT USING è una versione del PRINT che è estremamente utile per realizzare tabelle.
Immagina di volere un massimo di 6 cifre in ogni colonna della tabella. Per ottenerlo devi usare una linea come questa:

```
760 PRINT USING "######";NU
```

Questa linea arrotonderà il valore di NU (se ha un punto decimale) e aggiunge spazio prima del numero fino a quando la lunghezza totale dei caratteri e degli spazi è di 6 caratteri. Per es. se il valore di NU fosse 35.6 allora apparirebbe così (il simbolo '+' rappresenta uno spazio, ma non lo vedrai apparire sullo schermo).

```
++++37
```

Il numero è stato arrotondato e sono stati aggiunti 4 spazi, così con le due cifre (3 e 7) il numero totale dei caratteri è 6, che corrisponde esattamente al numero di cancellini (#) che hai usato nella istruzione PRINT USING. Se avessi usato 7 cancellini allora sarebbero stati aggiunti 5 spazi prima del numero.
Se il valore di NU fosse un numero contenente più di 6 cifre allora il computer mostrerebbe 6 stelle al posto del numero, indicando che il numero era troppo lungo.
Il cancellino può essere usato anche per definire quanti caratteri di una stringa vuoi che vengano mostrati. Per es. se digiti:

```
PRINT USING "####";"ABCDEFG"
```

allora solo i caratteri ABCD verranno mostrati perché hai detto al computer che vuoi solo 4 caratteri.
Puoi anche chiedere al computer di indicare se un numero è positivo o negativo e dove mettere il segno + o –. Per es. se digiti:

```
PRINT USING "+####";4325
```

allora il computer mostrerà il numero 4325 con un più davanti. Se il numero 4325 fosse stato negativo avrebbe avuto davanti un simbolo meno (prova cambiando +4325 in –4325).

Se ora digiti:

```
PRINT USING "####+";4325
```

allora vedrai che il computer mostra il numero 4325 con il simbolo *più* dietro.

Se provi i due es. precedenti con un segno meno invece del segno più, così:

```
PRINT USING "-####";4325
PRINT USING "####-";4325
```

e con un numero positivo e uno negativo, vedrai che se usi il segno meno allora il computer mostrerà un segno meno prima o dopo un numero (dipende dove hai detto al computer di metterlo) se il numero è negativo, ma non mostrerà un segno più se il numero è positivo.

È anche possibile dire al computer quanti numeri vuoi prima e dopo il punto decimale. Per es. se digiti:

```
PRINT USING "#####.####";143.65786
```

allora vedrai che il computer arrotonda il numero 143.65786 di 4 posti decimali e mette due spazi (rappresentati qui dal segno +) prima del numero, così:

```
++143.6579
```

Ciò succede perché abbiamo detto al computer che vogliamo 4 cifre dopo il punto decimale, così esso deve arrotondare il numero solo di poco. Gli abbiamo anche detto che vogliamo 5 cifre prima del punto decimale e dato che ne abbiamo solo 3 deve aggiungere due spazi.

Qualche volta è utile saper aggiungere delle virgole in un numero per maggior chiarezza. Per es. 1,000,000 è più chiaro di 1000000. Possiamo usare il comando PRINT USING per dire al computer di aggiungere virgole al posto giusto quando fa apparire un numero. Prova questo es.:

```
PRINT USING "##,###,###";1000000
```

il computer mostrerà il risultato

```
+1,000,000
```

se cambi il numero con 1000 allora il computer mostrerà

```
+++++1,000
```

Ciò succede perché abbiamo detto al computer di mostrare un totale di 7 caratteri (ecco perché vengono aggiunti degli spazi) senza contare le virgole e di aggiungere una virgola prima di ogni 3 cifre da sinistra (es. la virgola in 1,000 è prima della terza cifra da sinistra e la virgola in 1,000,000 è prima della sesta cifra da sinistra.

Il comando PRINT USING è utile anche quando si deve indicare una somma di denaro. Prova questo es.:

```
PRINT USING "$#######";1234
```

vedrai il risultato apparire sullo schermo

```
$+++1234
```

Comunque se digiti

```
PRINT USING "#$######";1234
```

avrai questo risultato:

```
+++$1234
```

Mettendo il simbolo del dollaro dopo il primo cancellino dici al computer che vuoi che il simbolo del dollaro sia fluttuante. Ciò significa che esso apparirà immediatamente a sinistra della prima cifra di un numero. Il primo cancellino viene sempre tenuto in considerazione dal computer (nell'es. sopracitato abbiamo un cancellino prima del simbolo del dollaro e sei dopo, e il computer li considera come 7 cancellini, come se il simbolo del dollaro non ci fosse). Se il simbolo del dollaro viene messo prima del cancellino, allora apparirà alla lontana sinistra del numero, in modo che qualsiasi spazio che il computer aggiunge andrà tra il simbolo del dollaro e il numero stesso.

Se è necessario mostrare un numero in forma esponenziale (ad es. 3E+04 è 3×10 alla potenza di 4 e 7E–02 è 7×10 alla potenza di –2), allora il comando PRINT USING può provvedere a ciò. Devi solo aggiungere 4 simboli alla fine della istruzione PRINT USING, in questo modo:

```
PRINT USING "##↑↑↑↑";14326
```

Questo comando mostrerà il risultato

```
1E+04
```

che è 1×10 alla potenza di 4, o 100000. Il computer ha arrotondato per difetto il numero 14326 in 10000. Se invece digiti:

```
PRINT USING "#↑↑↑↑↑";14326
```

avrai il risultato

```
14E-03
```

che è 14×10 alla potenza di 3, cioè 14000.

Un'altra utile caratteristica del comando PRINT USING è l'abilità di centrare una stringa in un campo. Se avessi una colonna in una tabella che fosse della larghezza di 6 caratteri e volessi centrare le lettere AA in quella colonna, allora il computer dovrà stampare due spazi, poi le AA e poi altri due spazi. Digita:

```
PRINT USING "######=";"AA"
```

allora il computer mostrerà:

```
++AA++
```

Naturalmente non potrai vedere realmente gli spazi (rappresentati dai segni +) dopo le lettere AA. Usando 6 cancellini nel comando PRINT USING dici al computer che vuoi che il campo sia della larghezza di 6 caratteri. Il simbolo = dice al computer che vuoi che i caratteri che seguiranno siano centrati.

Infine se vuoi stampare una stringa verso la destra del campo, allora userai un comando come questo:

```
PRINT USING "#####>";"AA"
```

con il quale otterrai questo risultato

```
+++AA
```

Hai detto cioè al computer di far apparire i caratteri nella parte destra del campo che è della larghezza di 5 caratteri.

Riassunto

Il comando PRINT USING può essere usato in molti modi per facilitare la stampa di tabelle e in generale per ordinare lo schermo. È possibile:

1) Definire quanti caratteri o quante cifre vuoi che appaiono sullo schermo (un cancellino rappresenta un carattere).

2) Definire se vuoi un simbolo più o meno (+ o –) dopo o davanti ad un numero (mettendo un simbolo + o – davanti o dopo i cancellini).

3) Specificare quante cifre vuoi prima e dopo il punto decimale (mettendo il punto decimale al giusto posto tra i cancellini).

4) Specificare dove necessitano le virgole in un numero (mettendolo al punto giusto tra i cancellini).

5) Dire al computer di mettere il simbolo della lira (o un altro) prima di un numero, sia fluttuante che fisso, mettendolo dopo (fluttuante) o prima (fisso) del primo cancellino.

6) Mostrare un numero in forma esponenziale aggiungendo 4 segni ↑ dopo i cancellini.

7) Centrare una stringa in un campo usando il simbolo = dopo i cancellini.

8) Mostrare una stringa alla estrema destra del campo usando il simbolo > dopo i cancellini.

Naturalmente, ogni volta che in questo capitolo abbiamo usato stringhe e numeri puoi usare delle variabili e ogni volta che abbiamo usato delle variabili puoi usare delle stringhe o dei numeri. Il comando PRINT USING può essere usato sia in modo diretto sia nei programmi.

PUDEF

Il comando PUDEF è usato per alterare il modo in cui il comando PRINT USING fa apparire i numeri ed i caratteri. Per es. se vuoi che il computer mostri delle stelle anziché degli spazi allora userai questo comando:

```
PUDEF "*"
```

Se poi usi il comando:

```
PRINT USING "#####";12
```

allora il computer mostrerà il risultato

```
***12
```

In qualche caso potresti desiderare un trattino anziché una virgola, e per ottenerlo devi usare:

```
PUDEF " -"
```

O, se vuoi una virgola decimale, invece del punto, dovrai usare

```
PUDEF " ."
```

Potresti volere un simbolo / invece della virgola decimale, — nel qual caso userai:

```
PUDEF " ,/"
```

La funzione più utile del comando PUDEF è quella che ti permette di ottenere il simbolo lira invece di quello del dollaro, così:

```
PUDEF " ,.£"
```

Puoi cambiare uno o tutti i simboli '(spazio)','(virgola)','(punto)' e 'S' con qualsiasi altro simbolo usando il comando PUDEF. Avrai già osservato che il primo carattere del comando PUDEF è il carattere che rimpiazzerà lo spazio, il secondo quello che sostituirà il punto decimale e l'ultimo carattere quello che sostituirà il simbolo del dollaro. Questo significa che per far ritornare tutto alla situazione normale dovrai usare:

```
PUDEF " ,.$"
```

Ma se vuoi cambiare il punto decimale con un segno '%' (percentuale) userai:

```
PUDEF " ,%"
```

con il segno '%' che occuperà il posto del punto decimale.

Grafici

Fino ad ora i tuoi programmi hanno avuto qualche limitazione perché non hanno usato i colori e le eccellenti capacità grafiche del tuo computer. Probabilmente saprai che esso può mostrare 121 colori e può produrre alcune figure molto belle ed ora è venuto il momento di vedere come si fa.

COLOR

Forse saprai che 'colore' in inglese è scritto COLOUR e non COLOR, ma dato che il tuo computer lavora con un linguaggio che è stato inventato negli Stati Uniti, usa anche lo 'spelling' americano per la maggior parte dei comandi e naturalmente ciò succede con la maggior parte dei computer in circolazione.

Il comando COLOR può essere usato per cambiare il colore allo schermo, allo sfondo e a tutto ciò che c'è sullo schermo. Se digiti:

```
COLOR 0,15,6
```

lo schermo diventerà tutto dello stesso colore.

Il primo numero del comando COLOR dice al computer di quale parte vuoi cambiare il colore. Il numero può essere da 0 a 5 e rappresenta:

0—il colore dello schermo;
1—il colore del testo;
2—multicolore 1;
3—multicolore 2;
4—il colore del bordo.

Non sai ancora cos'è il modo multicolore, ma quando lo incontrerai avrai bisogno di sapere che il comando COLOR viene usato per scegliere i colori.

Il secondo numero è il colore reale che vuoi. È un numero tra 1 e 16. 1 è nero, 2 è bianco, 3 è rosso e così via, il numero di ogni colore corrisponde al numero del tasto su cui è scritto il colore. I numeri dei colori sulla parte inferiore dei tasti si ottengono aggiungendo 8 al numero di quei tasti (ad es. il rosa ha il numero 12 che è il risultato di 8+4).

Il terzo numero è il livello di luminosità. È un numero che va da 0 a 7, con lo 0 che è il più scuro e il 7 che è il più chiaro. La luminosità non ha alcun effetto sul nero. Potendo scegliere 8 livelli di luminosità per 15 colori (il nero è escluso) se ne avranno 120, più il nero, cioè 121.

Per vedere lo spettro completo dei colori che il tuo computer può produrre, digita questo breve programma:

```
10 SCNCLR
20 FOR C=1 TO 16:FOR I=0 TO 7
30 COLOR 1,C,I:PRINT"[RVS ON] [RVS OFF]";
40 NEXT I,C
```

Questo programma cambierà il colore del testo, attraverso tutti i colori possibili e i vari livelli di luminosità e farà apparire un cubo colorato per ognuno di questi colori. Se vuoi vedere lo schermo o il bordo in tutte le possibili gradazioni di colore dovrai cambiare la linea 30 con:

```
30 COLOR 0,C,I:COLOR 4,C,I:FOR M=1 TO 500:NEXT M
```

GRAPHIC

Il comando GRAPHIC è usato per scegliere in quale modo grafico vuoi lavorare. Ci sono 5 differenti modi e ognuno ha i suoi vantaggi ed i suoi svantaggi. Questi modi sono:

GRAPHIC 0:	Testo normale sullo schermo.
GRAPHIC 1:	Schermo con alta-risoluzione di 320 punti in orizzontale e 200 in verticale.
GRAPHIC 2:	Simile al G.1 eccetto perché ha spazio per 5 linee di testo normale nella parte bassa dello schermo. La risoluzione è di 320 punti in orizzontale e di 180 in verticale.
GRAPHIC 3:	Grafico multicolore. Permette di usare più colori in una piccola area ed ha una risoluzione di 160 punti in orizzontale e 200 in verticale.
GRAPHIC 4:	Simile al G.3 eccettuato il fatto che ha 5 linee di testo normale. La risoluzione è di 160 punti in orizzontale e di 180 in verticale.

Un tipico comando grafico è:

```
10 GRAPHIC 2,1
```

Aggiungendo ',1' alla fine del comando GRAPHIC scegli il modo grafico e lo schermo si pulisce. Togliendo ',1' o mettendo ',0' alla fine si seleziona il modo grafico desiderato senza pulire lo schermo.

Usando i grafici si utilizza molta memoria, ma quando hai finito puoi accedere alla memoria di nuovo digitando:

```
GRAPHIC CLR
```

Questo comando permette di usare la memoria che il computer riserva per i grafici.

LOCATE

Lo schermo dell'alta risoluzione ha un cursore speciale chiamato cursore pixel, che a noi è invisibile, ma dice al computer esattamente dove disegnare sullo schermo. Sebbene noi non possiamo vedere questo cursore, possiamo muoverlo in ogni punto dello schermo. Per farlo dobbiamo dare al computer le coordinate orizzontali e verticali espresse in pixel, a partire dall'angolo in alto a sinistra, così:

```
175 LOCATE 40,30
```

Questa linea dice al computer di posizionare (LOCATE) il cursore pixel nel punto che è 40 pixel in orizzontale dalla parte sinistra dello schermo e a 30 pixel in verticale dalla parte superiore dello schermo.

Possiamo anche dire al computer di muovere il cursore pixel da un numero dato di pixel in su o in giù, a sinistra o a destra, così:

```
240 LOCATE +10,-5
```

Questa linea dice al computer di muovere il cursore pixel di 10 pixel a destra e di 5 pixel verso l'alto dalla posizione attuale. Se vogliamo che il cursore pixel si muova di 10 pixel a sinistra e di 5 pixel verso il basso, useremo una linea del genere:

```
255 LOCATE -10,+5
```

Puoi anche dire al computer di muovere il cursore pixel un certo numero di pixel in una direzione con un certo angolo. L'angolo viene misurato dalla verticale come con un rilevamento alla bussola.

```
354 LOCATE 40;45
```

Il cursore pixel si muoverà di 40 pixel con un angolo di 45 gradi nord-est usando l'analogia con la bussola.

DRAW

Il comando DRAW ti permette di disegnare sullo schermo. Puoi disegnare un singolo punto, una linea o parecchie linee (rette). Prova questo programma:

```
10 GRAPHIC 1,1:COLOR 0,1,7:COLOR 4,7,7:COLOR 1,2,7
20 DRAW 1,10,10
```

Quando userai questo programma vedrai apparire un punto bianco sullo schermo che a sua volta sarà nero con il bordo blu.

La linea 10 seleziona il modo grafico 1 e pulisce lo schermo. Poi seleziona il colore dello schermo (nero), del bordo (giallo) e del disegno (bianco).

Se guardi alla linea 20 vedrai che abbiamo dato al comando DRAW 3 numeri. Il primo è la fonte del colore. Può essere un numero qualsiasi da 0 a 3. Selezionando lo 0 dici al computer di disegnare con il colore dello sfondo e ottieni l'effetto di cancellare tutto ciò che disegni. Selezionando l'1 dici al computer di disegnare con il colore del testo attuale (che in questo caso è bianco). Selezionando il 2 dici al computer di disegnare in multicolore 1 e selezionando il 3 dici al

computer di disegnare in multicolore 2. Se il programma non usa il modo multicolore, non si potranno accostare più di due colori (vedi anche Grafici multicolore).

Il secondo numero indica la coordinata X (orizzontale) a partire da sinistra e il terzo numero la coordinata Y (verticale) a partire dall'alto; dalle coordinate X e Y il computer saprà da dove iniziare a disegnare. Abbiamo detto al computer che vogliamo iniziare a disegnare dal punto di coordinate 10, 10 (10 pixel orizzontale e 10 pixel verticale) e dato che non abbiamo detto al computer di disegnare qualcosa, esso accenderà soltanto un pixel in quel punto.

Se guardi il punto sullo schermo vedrai quanto è piccolo un pixel, perché infatti quel punto è un pixel. Quando pensi che (nel GRAPHIC 0) ci sono 320 pixel sullo schermo in orizzontale e 200 in verticale, ti accorgerai di come puoi disegnare una figura in modo particolareggiato.

Il pixel rimarrà sullo schermo fino a quando non dirai al computer di ritornare al testo normale sullo schermo. Il modo più facile per farlo è digitare un qualsiasi carattere (eccetto i numeri) e poi premere il RETURN. Lo schermo ritornerà normale e riceverai un messaggio d'errore, ma non succederà niente.

Se ora modifichi la linea 20 in questo modo:

```
20 DRAW 1,0,0 TO 319,199
```

ed esegui il programma vedrai che si disegnerà una linea bianca in diagonale attraverso lo schermo dall'angolo in alto a sinistra all'angolo in basso a destra.

Quello che ora abbiamo detto al computer è di disegnare una linea dal punto di coordinate 0,0 al (TO) punto di coordinate 319,199. Possiamo continare aggiungendo altro al comando DRAW per ottenere qualcosa come:

```
20 DRAW 1,0,0 TO 319,199 TO 319,0 TO 0,199 TO 0,0
```

Appariranno sullo schermo due triangoli con i vertici che si incontrano al centro.

Puoi anche dire al computer di disegnare una linea dal cursore pixel ad un punto qualsiasi. Per es. se cambi la linea 20 con:

```
20 LOCATE 10,10:DRAW 1 TO 50,50
```

ed esegui il programma, il cursore posizionerà il cursore pixel alle coordinate 10,10 e poi disegnerà una linea da quel punto al punto di coordinate 50,50 nel colore attuale del testo.

Puoi disegnare anche da un punto ad un altro, tanti pixel a sinistra o a destra dal punto di partenza e tanti pixel in su o in giù rispetto sempre al punto di partenza, così:

```
20 LOCATE 10,10:DRAW 1 TO +20,+10 TO -10,-5
```

Questa volta quando esegui il programma vedrai che il computer ha disegnato una linea del cursore pixel al punto a 20 pixel a destra e a 10 pixel in basso dal cursore pixel, e un'altra linea da questo punto al punto a 10 pixel a sinistra e a 5 pixel verso l'alto.

Il comando DRAW può anche disegnare le linee di un angolo. Per es. se cambi la linea 20 con:

```
20 LOCATE 50,50:DRAW 1 TO 40;32
```

ed esegui il programma, vedrai che il computer disegna una linea lunga 40 pixel dal cursore pixel con un angolo di 32 gradi.

BOX

Il comando BOX è usato per disegnare quadrati e rettangoli. Per vedere come si usa cambia la linea 20 del programma precedente con:

```
20 BOX 1,110,50,210,150
```

ed eseguendo il programma vedrai apparire un quadrato in mezzo allo schermo. Ciò succede perché abbiamo detto al computer di disegnare un rettangolo (BOX) con l'angolo in alto a sinistra alle coordinate 110,50 e l'angolo in basso a destra alle coordinate 210,150. Il computer studia ora in che punto devono essere gli altri vertici e disegna una scatola (il primo numero è la fonte di colore come nell'istruzione DRAW).

È possibile anche far ruotare la scatola. Per es. se aggiungi ',45' alla fine del comando BOX sulla linea 20, così:

```
20 BOX 1,110,50,210,150,45
```

ed esegui il programma, vedrai disegnato sullo schermo un quadrato ruotato di 45 gradi. L'angolo di rotazione può essere di qualunque ampiezza da 0 a 360 gradi (con un angolo di 360 o 0 gradi non c'è rotazione).

Puoi anche colorare la scatola, basta aggiungere ',1' dopo l'angolo di rotazione (o solo ',1' se non vuoi far ruotare la scatola) così:

```
20 BOX 1,110,50,210,150,45,1
```

oppure

```
20 BOX 1,110,50,210,150,,1
```

Il numero ',1' alla fine dell'istruzione BOX dice al computer di colorare la scatola con il colore attuale del testo (il colore di cui sono colorati i bordi del quadrato).

CIRCLE

Il comando CIRCLE è usato per disegnare le circonferenze, ma serve per disegnare anche ellissi, triangoli, quadrati e in pratica qualunque altra figura, come presto vedrai.

Se provi a cambiare la linea 20 del precedente programma con:

```
20 CIRCLE 1,160,100,80
```

vedrai apparire sullo schermo una circonferenza con raggio di 80 pixel, con il centro corrispondente al centro dello schermo.

Il primo numero è la fonte di colore. Il secondo e il terzo sono le coordinate X e Y del centro della circonferenza. L'ultimo numero è il raggio della circonferenza in pixel.

Se ora aggiungi un ',40' alla fine del comando CIRCLE, così:

```
20 CIRCLE 1,160,100,80,40
```

ed esegui il programma, verrà disegnata sullo schermo una elissi che è alta la metà di quanto è larga. Ciò avviene perché il numero 80 (il raggio) è in effetti il raggio orizzontale (raggio X), ovvero la metà della larghezza dell'ellisse in pixel. Il 40 che viene aggiunto è invece il raggio verticale (raggio Y) ovvero la metà dell'altezza dell'ellisse in pixel. Se la X e la Y sono uguali, allora la figura sarà una circonferenza, mentre se il raggio X è maggiore l'ellisse sarà più larga che alta. È anche possibile disegnare un ARCO.

Se modifichi la linea 20 come qui di seguito, allora otterrai il disegno di un arco:

```
20 CIRCLE 1,160,100,80,80,90,180
```

Gli ultimi due numeri che abbiamo aggiunto, dicono al computer l'angolo di partenza e di arrivo dell'arco. Il ',90' dice al computer di disegnare l'arco partendo da un angolo di 90 gradi mentre il ',180' dice al computer di finire l'arco con un angolo di 180 gradi.

È anche possibile ruotare la circonferenza e l'arco (potrebbe sembrare senza significato ma tra poco vedrai quanto può essere utile). Per farlo devi semplicemente aggiungere l'angolo di rotazione alla fine del comando CIRCLE, così:

```
20 CIRCLE 1,160,100,80,80,90,180,20
```

Il programma disegnerà lo stesso arco ma lo ruoterà di 20 gradi. Infine, puoi usare il comando CIRCLE per disegnare altre figure, come triangoli ed esagoni. Per farlo devi dire al computer quanti gradi vuoi tra ogni lato della figura (questo angolo deve essere di 360 gradi e diviso per il numero di lati della figura che

desideri). Per es. se aggiungi ',120' alla fine della linea 20 verrà disegnato un triangolo:

```
20 CIRCLE 1,160,100,80,80,0,0,0,120
```

SCALE

Il comando SCALE è semplice e può essere sia attivato che disattivato. Per attivare la facilitazione del comando SCALE devi digitare SCALE 1 e per disattivarla devi digitare SCALE 0. Se aggiungi questa linea al programma che abbiamo usato fino ad ora vedrai cosa fa esattamente questo comando:

```
15 SCALE 1
```

Quando ora esegui il programma vedrai apparire un triangolo nell'angolo in alto a sinistra sullo schermo e sarà molto più piccolo perché il comando SCALE fa in modo che il computer creda di avere lo schermo dell'ampiezza di 1024 pixel in orizzontale e 1024 pixel in verticale, anche se in effetti è di 320 pixel in orizzontale e 200 in verticale (nel modo normale di alta-risoluzione) o 160 in orizzontale e 200 in verticale (nel modo multicolore). Ciò può essere utile per i grafici.

PAINT

Il comando PAINT è usato per aggiungere un po' di colore alle tue figure. È molto semplice da usare, basta dire al computer quale fonte di colore desideri usare per riempire (PAINT) la figura. Cancella la linea 15, digita SCALE 0 (direttamente) e poi digita questa linea:

```
30 PAINT 1,160,100,1
```

e quando esegui il programma il tuo triangolo sarà colorato di bianco perché hai detto al computer di colorare quell'area del colore del testo partendo dal punto di coordinate 160,*100*. Il numero ',1' alla fine dice al computer che vuoi che smetta di colorare quando raggiunge una linea che è di colore diverso rispetto allo sfondo. Se tralasci il numero '1' allora il computer continuerà a colorare fino a quando non trova una linea di confine dello stesso colore con cui sta colorando.

Per scegliere quale colore usare devi cambiare il colore del testo con quello con cui vuoi colorare. Per colorare il tuo triangolo di rosso, ad es., devi cambiarne il colore del testo così:

```
25 COLOR 1,3,4
```

Grafici multicolore

Se hai fatto un po' di esperimenti con i tuoi grafici avrai notato che non puoi avere più di due colori in una singola casella di 8 × 8 pixel (ti ricordi che lo schermo è diviso in caselle? Se non lo ricordi, ritorna al capitolo con il comando CHAR). Se digiti questo breve programma:

```
10 GRAPHIC 1,1:COLOR 0,1,7:COLOR 4,1,7:COLOR
   1,2,7
20 CIRCLE 1,50,50,20:PAINT 1,50,50
30 COLOR 1,5,5
40 CIRCLE 1,70,70,25:PAINT 1,70,70
50 COLOR 1,6,5
60 CIRCLE 1,40,90,15:PAINT 1,40,90
```

vedrai delle circonferenze disegnate sullo schermo e poi colorate. Vedrai anche che dove le circonferenze oltrepassano la casella cambiano di colore, perché non puoi avere più di due colori in una casella carattere e uno di questi è il colore dello sfondo. Per risolvere questo problema devi usare uno dei modi grafici multicolore. Prova cambiando il tuo programma con:

```
10 GRAPHIC 3,1:COLOR 0,1,7:COLOR 4,1,7:COLOR
   1,2,7
20 CIRCLE 1,50,50,20:PAINT 1,50,50
30 COLOR 2,5,5
40 CIRCLE 2,70,70,25:PAINT 2,70,70
50 COLOR 3,6,5
60 CIRCLE 3,40,90,15:PAINT 3,40,90
```

Questa volta quando esegui il programma verranno disegnate 3 differenti circonferenze, verranno colorate e ognuna sarà del suo colore in ogni suo punto! Ciò succede perché stai usando il modo multicolore con il quale si possono avere fino a 4 colori per una singola posizione (essendo uno di questi colori quello dello sfondo).

Osserverai che per ogni circonferenza usiamo una differente fonte di colore, definita dal comando COLOR. La seconda circonferenza è disegnata in multicolore 1 (che è di fonte di colore 2) e la terza circonferenza è disegnata in multicolore 2 (che è la fonte di colore 3). La prima circonferenza è ancora colorata con la fonte di colore 1 (la fonte di colore normale del testo).

Avrai comunque notato che le circonferenze appaiono un po' più spesse, perché quando si usa il multicolore, la risoluzione orizzontale si riduce, così invece di avere 320 pixel in orizzontale sullo schermo, ne abbiamo solo 160. Perciò per delle figure molto dettagliate, che non hanno bisogno di molti colori,

è meglio usare il GRAPHIC 1 o il GRAPHIC 2, ma dove il colore è predominante è meglio usare il GRAPHIC 3 o il GRAPHIC 4.

Una cosa speciale che possiamo fare con la fonte di colore 3 quando usiamo in modo multicolore, è di cambiare il colore di tutto ciò che è stato disegnato. Per es. se hai disegnato una circonferenza usando la fonte di colore 3 che è stata inserita sul bianco, allora verrà disegnata una circonferenza bianca. Se poi cambi il colore della fonte di colore con il rosso, allora la circonferenza che hai appena disegnato, diventerà subito rossa. Prova ad aggiungere questo al tuo programma:

```
70 FOR C=1 TO 8:FOR I=0 TO 7
80 COLOR 3,C,I:FOR M=1 TO 500:NEXT M
90 NEXT I,C
```

Ora quando lo esegui vedrai una delle circonferenze (quella disegnata con la fonte di colore 3) cambiare in tutti i differenti colori e luminosità possibili sul tuo computer.

SSHAPE e GSHAPE

SSHAPE e GSHAPE sono due comandi grafici molto utili perché ti permettono di mettere da parte un'area dello schermo in una variabile stringa e poi utilizzarla in qualche altro posto. Ciò significa che puoi muovere facilmente una figura complessa, come una navicella spaziale.

Il comando SSHAPE è quello che tiene da parte l'area dello schermo in una variabile stringa. Tutto ciò che devi fare è dire al computer in quale stringa vuoi che venga messa la figura e dare le coordinate dell'angolo in alto a sinistra e dell'angolo in basso a destra della figura. Ecco un esempio:

```
320 SSHAPE SP$,40,50,60,70
```

Questa linea immagazzinerà l'area di memoria con l'angolo in alto a sinistra alle coordinate 40,50 e l'angolo in basso a destra alle coordinate 60,70, nella variabile stringa SP$.

Per rimettere la figura di nuovo sullo schermo non dovrai far altro che dire al computer in quale variabile è stata immagazzinata la figura e dare le coordinate dell'angolo in alto a sinistra della figura, così:

```
400 GSHAPE SP$,60,70
```

Questa linea metterà la figura immagazzinata nella variabile stringa SP$ di nuovo sullo schermo con l'angolo in alto a sinistra della figura al punto di coordinate 60,70.

Se digiti questo breve programma vedrai una palla muoversi lentamente sullo schermo e che salta fuori dai limiti (sfortunatamente queste figure si muovono molto lentamente).

```
10 COLOR 1,6,5:COLOR 0,2,7:COLOR 4,3,4:GRAPHI
   C 1,1
30 CIRCLE 1,160,100,3:PAINT 1,160,100
40 SSHAPE PALLA$,156,96,164,104
50 X=160:Y=100:XD=1:YD=1
60 GSHAPE PALLA$,X-4,Y-4
70 X=X+XD:Y=Y+YD
80 IF X>315 THEN XD=-1
90 IF X<4 THEN XD=1
100 IF Y>195 THEN YD=-1
110 IF Y<4 THEN YD=1
120 GOTO 60
```

Ecco la spiegazione di come funziona il programma:

Linea 10: Stabilisce il verde quale colore del testo, il bianco per lo sfondo e il rosso per il bordo, poi seleziona il modo grafico 1 e pulisce lo schermo.
Linea 30: Disegna una circonferenza con il colore attuale del testo con un raggio di 3 pixel con il centro alle coordinate 160,*100* e poi la colora del colore attuale del testo.
Linea 40: Immagazzina la figura con l'angolo in alto a sinistra alle coordinate 156,96 e l'angolo in basso a destra alle coordinate 164,104 nella variabile stringa PALLA$.
Linea 50: Assegna alla variabile X il valore di 160, alla variabile Y il valore 100 e alla variabile XD e YD il valore 1.
Linea 60: Mette la figura immagazzinata nella variabile stringa PALLA$ con l'angolo in alto a sinistra al punto di coordinate X-3 e Y-3.
Linea 70: Aggiunge il valore della variabile XD alla variabile X e il valore di YD alla variabile Y.
Linea 80: Se il valore di X è maggiore di 315 assegna il valore –1 alla variabile XD.
Linea 90: Se il valore di X è minore di 4, assegna il valore 1 alla variabile XD.
Linea 100: Se il valore di Y è maggiore di 195, assegna il valore –1 alla variabile YD.
Linea 110: Se il valore di Y è minore di 4, assegna il valore 1 alla variabile YD.
Linea 120: Va alla linea 60 e continua il programma da quel punto.

In questo momento stiamo mettendo la figura sullo schermo, esattamente com'era quando per la prima volta l'avevamo immagazzinata nella variabile

stringa PALLA$. Comunque possiamo far ritornare la figura sullo schermo in modo differente. Per vedere come si può fare, modifica la linea 60 con:

```
60 GSHAPE PALLA$,X-4,Y-4,1
```

Quando esegui il programma, vedrai la palla muoversi sullo schermo, come prima, ma questa volta è in negativo. Cioè se prima la palla era verde su sfondo bianco, adesso è bianca su sfondo verde.

Se ora cambi la linea 60 con:

```
60 GSHAPE PALLA$,X-4,Y-4,2
```

ed esegui il programma vedrai una linea verde con la parte frontale ricurva muoversi sullo schermo. È ancora la palla ma questa volta la forma della palla è stata visualizzata in modo OR (oppure) con lo sfondo. Il computer cioè confronta ogni punto della figura con il punto dello schermo che sarà occupato; se uno o l'altro o entrambi i punti sono accesi, allora il computer accenderà il punto quando mette la figura sullo schermo. Ciò significa che puoi mettere una figura sopra l'altra, come se una si muovesse sull'altra. Per es. se sovrapponi con il modo OR la figura "|" con la figura "—", otterrai questo risultato: +.

Ora cambia la linea 60 così e guarda cosa ottieni:

```
60 GSHAPE PALLA$,X-4,Y-4,3
```

Questa volta quando esegui il programma vedrai che la palla è sparita e non riapparirà più. Se, però, aggiungi questa linea:

```
20 COLOR 1,3,4:BOX 1,0,150,319,170,,1:COLOR 1,6,5
```

e fai girare il programma, vedrai una banda rossa attraversare la parte inferiore dello schermo, e quando la palla invisibile si muoverà, vedrai parte della banda rossa cambiare in verde e un sentiero bianco taglierà di traverso la banda. Non puoi vedere la palla perché è stata visualizzata in modo AND (e) con lo sfondo. Ciò significa che quando il computer mette la figura sullo schermo confronta ogni punto della figura con il punto sullo schermo che esso occuperà; se entrambi sono accesi il computer accenderà il pixel dello schermo. Per es. se sovrapponi in modo AND la figura "+" con la figura "|" il risultato sarà la figura "|" (perché è l'unica area che le due figure hanno in comune).

Infine se cambi la linea 60 con:

```
60 GSHAPE PALLA$,X-4,Y-4,4
```

e fai girare il programma vedrai la palla muoversi sullo schermo lasciando una striscia dietro di sé perché ogni punto della figura è stato visualizzato in modo

XOR (OR esclusivo). XOR è molto simile ad OR, eccettuato il fatto che un punto si illumina se è acceso sulla figura o sullo schermo, ma non se è acceso su entrambi. Ciò significa che XOR "|" con "—" da "+" ma XOR "|" con "|" dà " ", cioè niente.

RCLR

RCLR è una funzione ed è usata per sapere quale colore è assegnato ad una fonte di colore. Per es. se abbiamo una linea come questa in un programma:

```
320 Z=RCLR(0)
```

allora alla variabile Z verrà dato il valore del colore attuale dello sfondo. Il numero della fonte di colore (tra parentesi) va dallo 0 al 4, proprio come per il comando COLOR.

RLUM

RLUM è un'altra funzione, simile alla RCLR tranne per il fatto che è usata per trovare il livello della luminosità attuale della fonte di colore. È usata nello stesso modo della funzione RCLR, infatti la linea

```
1550 CZ=RLUM(4)
```

assegnerà alla variabile C2 il valore della luminosità attuale del bordo.

RGR

È un'altra funzione grafica e si usa per sapere qual è il modo grafico attuale, infatti la linea

```
30 F=RGR(0)
```

assegnerà alla variabile F il valore del modo grafico attuale (0-4). Il numero tra parentesi può essere un numero qualsiasi perché è di poca importanza (in altre parole il computer lo ignora).

RDOT

RDOT è ancora un'altra funzione grafica e serve per avere alcune informazioni sul cursore pixel e può essere usata in questi modi:

```
100 Z=RDOT(0)
```

assegna alla variabile Z l'attuale coordinata X del cursore pixel;

```
100 PRINT RDOT(1)
```

mostra l'attuale coordinata Y del cursore pixel;

```
100 AA=RDOT(2)
```

assegna alla variabile AA il valore dell'attuale fonte di colore.

Nelle prossime pagine c'è il LISTato del programma ARTISTA che ti permetterà di disegnare delle figure sullo schermo in qualsiasi modo grafico. Sarebbe una buona idea digitarlo e leggere la spiegazione sul suo funzionamento; ti potrebbe aiutare a capire i comandi grafici.

Programma ARTISTA

Questo programma usa in modo completo le capacità grafiche del computer per permetterti di disegnare delle figure sullo schermo. Puoi disegnare circonferenze, triangoli, quadrati, linee rette, segmenti cioè praticamente qualunque cosa.

Quando eseguirai il programma ti verrà chiesto per prima cosa con quale modo grafico vuoi lavorare. Dovresti rispondere con un numero tra 1 e 4. Così lo schermo si pulisce e sia il bordo che lo schermo diventeranno neri e in mezzo allo schermo apparirà una piccola croce rossa; è il tuo cursore che puoi muovere su tutto lo schermo usando i tasti controllo cursore. Volendo puoi cambiare il colore dello schermo premendo la S e poi il colore desiderato (da 1 a 8). Lo schermo si pulisce (non farlo se hai sullo schermo una figura che vuoi tenere) e diventerà del colore che hai scelto. Anche il colore del bordo può essere cambiato semplicemente premendo la E e poi il colore desiderato.

Puoi cambiare anche il colore del disegno (e cambierà anche il colore del cursore) premendo la X e poi il colore desiderato. Se vuoi cambiare la luminosità, basta premere la I e poi la luminosità che desideri; non avrà effetto mentre cambi il colore di qualcosa. Per es., se cambi la luminosità con 7, poi cambi il colore del bordo con rosso, allora il bordo diventerà di un rosso brillante, ma tutto il resto dello schermo rimarrà dello stesso valore di luminosità.

La più *semplice possibilità* del programma ARTISTA è quella per il disegno. Se premi la D e poi muovi il cursore sullo schermo vedrai disegnarsi una linea mentre ti muovi. Premendo la D cancellerai questa funzione.

Puoi disegnare anche delle circonferenze o qualsiasi altra figura che di solito puoi disegnare con il comando CIRCLE. Per farlo devi posizionare il cursore sullo schermo dove vuoi che sia il centro della tua circonferenza e poi premere la C; lo schermo si pulirà, ma non preoccuparti, la tua figura non è andata persa. Ti

verranno poi chieste alcune informazioni sulla tua circonferenza: il raggio X e Y, l'angolo di partenza e di arrivo, l'angolo di rotazione e il numero di gradi di differenza tra ogni segmento della circonferenza. Premendo soltanto il RETURN in risposta a queste domande, si inserirà il valore sullo 0 o sul 2 nel caso della domanda su 'i gradi di differenza tra ogni segmento'. Una volta che hai introdotto questi valori la circonferenza verrà disegnata per te.

Se ti sei stancato dei movimenti lenti del cursore potresti premere la F ed esso si muoverà di 5 pixel per volta anziché di uno solo. Premendo la F una seconda volta il movimento del cursore ritornerà normale. Se usi la possibilità di disegnare mentre sei nel modo veloce, allora apparirà un segmento. Puoi disegnare anche dei rettangoli, basta prima posizionare il cursore nel punto in cui vuoi che sia un angolo del rettangolo (non è importante quale angolo sia) e poi premere la O (sta per Origine); apparirà un punto e tu dovrai muovere il cursore verso l'angolo opposto del tuo rettangolo e premere la B per farlo disegnare.

La possibilità ORIGINE è usata anche quando vuoi unire due punti sullo schermo. Se muovi il cursore sul primo punto e premi la O, poi muovi il cursore sul secondo punto e premi la J (JOIN), allora si disegnerà una linea tra i due punti. Muovendo il cursore su un altro punto e premendo di nuovo la J si disegnerà una linea dall'ultimo punto (dove hai premuto la J per la prima volta) fino a dove è posizionato il cursore. Puoi comunque muoverti in ogni punto dello schermo e premendo la J verrà disegnata una linea da dove si trova il cursore verso dove era il cursore quando avevi premuto la J la volta precedente.

Naturalmente puoi colorare un'area, posizionando il cursore in qualsiasi punto all'interno dell'area che vuoi colorare, assicurandoti di avere premuto il tasto colore giusto, e premendo la P; l'area si colorerà.

Quando stai usando uno dei modi multicolore avrai bisogno di selezionare differenti fonti di colore. Per cambiare il colore del multicolore 1 dovrai premere i due punti (:) e poi il colore che desideri (da 1 a 8). Per cambiare il colore del multicolore 2 dovrai premere il punto e virgola (;) e poi il colore che desideri. Fatto ciò puoi scegliere quale fonte di colore vuoi usare premendo semplicemente l'1 per un colore normale del testo, il 2 per il multicolore 1 e il 3 per il multicolore 2.

Nota: Se vuoi usare questo programma su un C16 dovrai togliere tutte le istruzioni REM e tutti gli spazi tra le istruzioni. Per es. se vedi queste due linee:

```
80 REM METTE UN BLOCCO VUOTO IN BK$
90 SSHAPE BK$,X-3,Y-3,X+3,Y+3
```

devi eliminare la linea 80 e cambiare la linea 90 con:

```
90 SSHAPEBK$,X-3,Y-3,X+3,Y+3
```

Se non lo farai, la memoria non sarà sufficiente.

Inoltre se vedi qualcosa tra parentesi quadre (ad es. IF A$="[CR]") devi premere il tasto indicato tra parentesi (in questo caso il tasto con la freccia verso destra).
Infine alcune istruzioni vanno messe su di un'unica linea (per es. quelle di numero 30 e 40).

Come funziona il programma

ARTISTA è un programma molto lungo perciò non mi soffermerò a spiegare dettagliatamente come funziona, ma darò solo uno sguardo a ciò che ogni sezione ti aiuta a fare.
La linea 10 seleziona il colore dello schermo, del bordo e del testo, poi la linea 20 assegna dei valori ad alcune variabili che verranno usate nel programma. La linea 30 pulisce lo schermo e ti chiede quale modo grafico desideri. La tua risposta verrà controllata dalla linea 40 per assicurarsi che sia un modo corretto. Se scegli un modo che non va bene la domanda viene rifatta.
La linea 50 calcola quanti pixel ci sono in orizzontale sullo schermo nel modo che hai scelto e poi la linea 60 assegna alla variabile Y il numero di pixel in verticale. La linea 70 seleziona il modo grafico corretto e pulisce lo schermo.
La linea 90 immagazzina il contenuto dello schermo intorno al cursore (che è soltanto un'area vuota perché lo schermo è stato pulito) nella variabile BK$ e le linee 110-120 disegnano il cursore sullo schermo. Il cursore viene immagazzinato nella variabile CU$ dalla lina 140, prima che lo schermo venga pulito di nuovo con la linea 160. Il cursore che è stato appena immagazzinato nella variabile CU$, viene messo sullo schermo dalla linea 180.
Le linee 190-300 scandiscono la tastiera e saltano a varie sottoprocedure che cambiano il bordo dello schermo e i colori, disegnano circonferenze ed eseguono altre svariate figure. Le linee 320-350 controllano il cursore. La agevolazione DRAW viene inserita e disinserita dalla linea 370, e la linea 390 inserisce l'origine. La agevolazione JOIN viene eseguita dalle linee 410-420, mentre la linea 440 inserisce l'agevolazione BOX. Se hai premuto la F la linea 460 inserisce o disinserisce il movimento veloce e la linea 480 fa muovere il cursore di 5 pixel anziché di uno solo.
È importante assicurarsi che il cursore non vada fuori dallo schermo ed infatti le linee 500 e 510 controllano che questo non succeda mentre il cursore è in movimento. La linea 530 cancella il cursore riportando sullo schermo quello che c'era precedentemente in quell'area, prima che il cursore venga mosso dalla linea 560. L'area sullo schermo che sta per essere occupata dal cursore, viene immagazzinata nella variabile BK$ in modo che il cursore non cancelli ogni cosa sotto di sé.
La linea 590 disegna un punto sullo schermo se la facilitazione DRAW è inserita (o se è stata inserita l'origine, nel qual caso il punto indicherà l'origine). La linea 600 azzera le variabili Q e P se il punto che è stato appena disegnato deve indicare l'origine.

Se hai bisogno di disegnare un rettangolo, lo farai con la linea 620. Questa procedura rimuove prima di tutto il cursore, poi disegna la scatola e immagazzina l'area che sarà sotto il cursore (cioè un angolo della scatola) nella variabile BK$. La linea 630 manda indietro il programma alla linea 170 per far ritornare il cursore sullo schermo e incominciare a controllare i tasti di nuovo.

Le linee dalla 650 alla 750 sono sottoprocedure che cambiano i vari colori e la luminosità. Le linee 770-860 chiedono la dimensione della circonferenza e poi la disegnano, e le linee 880 e 890 colorano l'area del colore desiderato. Il colore dello schermo viene cambiato dalle linee 910-930 ed i due multicolore vengono scelti dalle linee 940-1010. La linea 1020 esamina la tastiera e assegna alla variabile T un valore numerico di qualunque carattere che sia stato digitato (il comando VAL converte una stringa in un numero).

Artista

```
10 GRAPHIC 0:COLOR 0,1,7:COLOR 4,1,7:COLOR 1,
   2,7
20 C=2:I=7:S=1:Z=4:V=5
30 SCNCLR:INPUT"QUALE MODO GRAFICO DESIDERI";
   GM
40 IF GM<1 OR GM>4 THEN30
50 IF GM>2 THEN WD=160:X=80:ELSE WD=320:X=160
60 Y=100
70 GRAPHIC GM,1
80 REM METTE UN BLOCCO VUOTO IN BK$
90 SSHAPE BK$,X-3,Y-3,X+3,Y+3
100 REM DISEGNA IL CURSORE
110 DRAW S,X-3,Y TO X-1,Y:DRAW S,X+1,Y TO X+3
    ,Y
120 DRAW S,X,Y-3 TO X,Y-1:DRAW S,X,Y+1 TO X,Y
    +3
130 REM METTE IL CURSORE IN CU$
140 SSHAPE CU$,X-3,Y-3,X+3,Y+3
150 SCNCLR
160 REM METTE IL CURSORE SULLO SCHERMO
170 GSHAPE CU$,X-3,Y-3,4
180 REM ASPETTA IL COMANDO
190 GETKEY A$
200 IF A$="1" THEN S=1
210 IF A$="2" THEN S=2
220 IF A$="3" THEN S=3
230 IF A$="X" THEN GOSUB 650
240 IF A$="I" THEN GOSUB 690
250 IF A$="E" THEN GOSUB 730
260 IF A$="C" THEN GOSUB 770
```

```
270 IF A$="P" THEN GOSUB 880
280 IF A$="S" THEN GOSUB 910
290 IF A$=":" THEN GOSUB 950
300 IF A$=";" THEN GOSUB 990
310 REM CONTROLLO CURSORE
320 IF A$="[CR]" THEN XD=1
330 IF A$="[CL]" THEN XD=-1
340 IF A$="[CU]" THEN YD=-1
350 IF A$="[CD]" THEN YD=1
360 REM ATTIVA/DISATTIVA IL DRAW
370 IF A$="D" AND P=0 THEN P=1:ELSE IF A$="D"
     THEN P=0
380 REM DEFINISCE L'ORIGINE
390 IF A$="O" THEN OX=X:OY=Y:P=1:Q=1
400 REM UNISCE L'ORIGINE COL CURSORE
410 IF A$="J" THEN GSHAPE BK$,X-3,Y-3:DRAW S,
    OX,OY TO X,Y:OX=X:OY=Y
420 IF A$="J" THEN SSHAPE BK$,X-3,Y-3,X+3,Y+3
430 REM ATTIVA LA FACILITAZIONE BOX
440 IF A$="B" THEN B=1
450 REM ATTIVA/DISATTIVA MOVIMENTO VELOCE
460 IF A$="F" AND F=0 THEN F=1:ELSE IF A$="F"
     THEN F=0
470 REM MUOVE IL CURSORE DI 5 PIXEL SE E' ATT
    IVO IL MOVIMENTO VELOCE
480 IF F=1 THEN XD=XD*5:YD=YD*5
490 REM CONTROLLA CHE IL CURSORE STIA NELLO S
    CHERMO
500 IF X+XD<0 OR X+XD>WD THEN XD=0
510 IF Y+YD<0 OR Y+YD>200 THEN YD=0
520 REM TOGLIE IL CURSORE
530 GSHAPE BK$,X-3,Y-3
540 REM MUOVE IL CURSORE
550 X=X+XD:Y=Y+YD:XD=0:YD=0
560 REM METTE IN BK$ L'AREA CHE VERRA' OCCUPA
    TA DAL CURSORE
570 SSHAPE BK$,X-3,Y-3,X+3,Y+3
580 REM DISEGNA UN PUNTO SULLO SCHERMO
590 IF P=1 THEN GSHAPE BK$,X-3,Y-3:DRAW S,X,Y
    :SSHAPE BK$,X-3,Y-3,X+3,Y+3
600 IF Q=1 THEN Q=0:P=0
610 REM DISEGNA UN RETTANGOLO TRA L'ORIGINE E
     IL CURSORE
620 IF B=1 THEN GSHAPE BK$,X-3,Y-3:BOX 1,OX,O
    Y,X,Y:SSHAPE BK$,X-3,Y-3,X+3,Y+3:B=0
630 GOTO 170
640 REM CAMDIA IL COLORE
650 GOSUB 1020:C=T
660 IF C<1 OR C>8 THEN 650
670 COLOR 1,C,I:RETURN
680 REM CAMBIA LUMINOSITA'
690 GOSUB 1020:I=T
700 IF I<0 OR I>7 THEN 690
710 COLOR 1,C,I:RETURN
720 REM CAMBIA IL COLORE DEL BORDO
730 GOSUB 1020:B=T
740 IF B<1 OR B>8 THEN 730
750 COLOR 4,B,I:RETURN
```

```
760 REM CHIEDE LE DIMENSIONI DEL CERCHIO
770 GRAPHIC 0,1:COLOR 1,2,7
780 INPUT"RAGGIO X";XR:INPUT"RAGGIO Y";YR
790 INPUT"ANGOLO D'INIZIO";SA:INPUT"ANGOLO DI
     FINE";EA
800 INPUT"ANGOLO DI ROTAZIONE";RA:INPUT"ANGOL
    O TRA I SEGMENTI";DB
810 GRAPHIC GM
820 IF DB=0 THEN DB=2
830 REM DISEGNA IL CERCHIO
840 COLOR 1,C,I
850 CIRCLE S,X,Y,XR,YR,SA,EA,RA,DB
860 RETURN
870 REM RIEMPIE UN'AREA
880 GSHAPE BK$,X-3,Y-3
890 PAINT S,X,Y,1:SSHAPE BK$,X-3,Y-3,X+3,Y+3:
    RETURN
900 REM CAMBIA IL COLORE DELLO SCHERMO E LO P
    ULISCE
910 GOSUB 1020:Q=T
920 IF Q<1 OR Q>8 THEN 910
930 COLOR 0,Q,I:SCNCLR:RETURN
940 REM CAMBIA IL MULTICOLORE 1
950 GOSUB 1020:Z=T
960 IF Z<1 OR Z>8 THEN 950
970 COLOR 2,Z,I:RETURN
980 REM CAMBIA IL MULTICOLORE 2
990 GOSUB 1020:V=T
1000 IF V<1 OR V>8 THEN 990
1010 COLOR 3,V,I:RETURN
1020 GETKEY A$:T=VAL(A$):RETURN
```

CAPITOLO 10

FUNZIONI

DEF FN

Dovresti ormai sapere che una funzione è un comando che prende una variabile e fa qualcosa con essa prima di dare una risposta. Ti sarai chiesto spesso se «non sarebbe stato utile avere una funzione che potesse...». Bene, fortunatamente è possibile costituire le proprie funzioni usando il comando DEFine FuNction (definizione di funzione). Per es. se hai bisogno di eseguire questa operazione parecchie volte

```
((ZZ/5)*32)↑2
```

puoi definire la funzione che eseguirà questa operazione con una linea del tipo:

```
10 DEF FN A1(ZZ)=((ZZ/5)*32)↑2
```

Ciò dice al computer di definire una funzione chiamata FNA1 che dividerà un numero (il numero è rappresentato da ZZ) per 5, moltiplica il risultato per 32 e poi eleva al quadrato il risultato. Per vedere come funziona, prova ad aggiungere queste linee e fai girare il programma:

```
20 SCNCLR
30 INPUT"DIGITA UN NUMERO QUALSIASI";N
40 PRINT"((";N;"/5*32)↑2=";
50 PRINT FNA1(N)
60 GOTO 30
```

Come vedrai, tutto quello che si deve fare per usare la nuova funzione è dare un numero (che viene chiuso tra parentesi) e il computer eseguirà una operazione usando quel numero. Il computer sostituirà il valore ZZ nella originale definizione di funzione con il numero che diamo alla funzione.

Puoi definire quante funzioni vuoi, ma devi dare ad ognuna un nome differente. Il nome della funzione è FN e qualsiasi combinazione di lettere e numeri va bene, a patto che questa combinazione non inizi con un numero o contenga un comando (proprio come le variabili). Per es. i nomi della funzioni FN1A e FNPRINTER non verranno accettati dal computer.

Le funzioni che puoi definire possono eseguire una qualsiasi operazione, ma possono contenere solo numeri, variabili numeriche o variabili intere. Le variabili stringa e i caratteri tra virgolette non possono essere usati nella funzione che hai definito.

Tasti funzione

Avrai sicuramente notato che i tasti funzione sono indicati da F1 a F7. A questi tasti sono stati assegnati dei comandi e per vedere quali sono devi digitare:

```
KEY
```

Il computer mostrerà così un elenco di ciò che fa ogni tasto.

Il comando KEY può essere usato anche per ridefinire ciò che fa ogni tasto. Per es. se vuoi ridefinire il tasto funzione 1 in modo che pulisca lo schermo e LISTi il programma quando viene premuto, allora dovrai digitare:

```
KEY 1,"SCNCLR:LIST"+CHR$(13)
```

Come puoi vedere i due comandi SCNCLR e LIST, sono tra virgolette e sono separati dai due punti. +CHR$(13) dice al computer che vuoi che i comandi vengano eseguiti immediatamente appena il tasto viene premuto; (CHR$(13) è il codice per il tasto RETURN e il computer risponde come se fosse stato premuto il tasto RETURN).

Qualche volta avrai bisogno di includere qualcosa tra virgolette e ciò potrà causare dei problemi. Per risolverli usiamo il comando CHR$(34) così:

```
KEY 3,"PRINT "+CHR$(34)+"CIAO"+CHR$(34)+CHR$(13)
```

Come puoi vedere ogni volta che hai bisogno delle virgolette, devi chiuderle e aggiungere +CHR$(34)+ poi aprire le virgolette e continuare con il resto del comando.

Potrai ridefinire uno qualsiasi dei tasti funzione, incluso il tasto HELP in modo che se stai scrivendo un programma nel quale hai bisogno di tasti speciali per realizzare qualcosa di speciale, puoi ridefinirne uno o tutti, per adattarli alle tue necessità.

Funzioni numeriche

La maggior parte delle funzioni numeriche disponibili sul tuo computer sono già state utilizzate, ma quelle di cui non abbiamo parlato, sono spiegate in ordine alfabetico nelle pagine seguenti.

ABS

Il comando ABS è usato per trovare il valore assoluto di un numero. Digita:

```
PRINT ABS(-64)
```

il computer mostrerà come risultato 64. Un numero che è già positivo rimarrà positivo, così:

```
PRINT ABS(53)
```

darà 53 come risposta.

DEC

La funzione decimale convertirà un numero esadecimale (base sedici) in decimale, così:

```
PRINT DEC("F2")
```

e darà il risultato di 242.

EXP

La funzione EXP (esponenziale) darà la costante matematica 'e' (che è la base dei logaritmi naturali ed ha approssimativamente il valore di 2.71828183) alla potenza di un qualsiasi numero. Per es.:

```
PRINT EXP(2)
```

eleverà e alla potenza di 2, dando il risultato di 7.3890561. Questa funzione particolare non è usata spesso, perciò non preoccuparti se non hai mai sentito parlare di 'e' e dei logaritmi.

LOG

È l'inverso di EXP ed è usato per trovare i logaritmi naturali.

```
PRINT LOG(5)
```

Darà il risultato 1.60943791

SGN

Se qualche volta hai bisogno di sapere se un numero è positivo o negativo, allora userai la funzione SGN. Essa ti darà il risultato –1 se il numero è negativo, 1 se il numero è positivo e 0 se il numero è zero. Ecco un esempio:

```
PRINT SGN(-54);SGN(0);SGN(992)
```

SQR

Questa funzione viene usata per trovare la radice quadrata di un numero. Prova:

```
PRINT SQR(169)
```

e avrai il risultato di 13.

USR

Il comando USR è usato per far partire un programma in linguaggio macchina passandogli un numero (quello tra parentesi). Per un uso completo di questa funzione, vedi il capitolo sul monitor.

Funzioni trigonometriche

Il tuo computer ha 4 funzioni trigonometriche –SIN, COS, TAN e ATN che, come ti aspetterai calcolano il seno, il coseno, la tangente di un angolo e la arcotangente di una tangente. Se non hai familiarità con queste funzioni non devi preoccuparti; probabilmente non avrai bisogno di usarle spesso e in qualsiasi caso, esse vengono spiegate dettagliatamente in una grande quantità di libri di

matematica. I comandi grafici che abbiamo già visto eseguono molte delle applicazioni per le quali dovresti usare le funzioni trigonometriche.

L'unico punto importante da ricordare, a proposito delle funzioni trigonometriche, è che gli angoli sono misurati in radianti. Un radiante è pari a 57.2957805 gradi e ci sono 2 PI radianti in una rotazione completa (360°). Potrai trovare comunque una spiegazione più completa del perché i matematici usano più i radianti dei gradi, in un qualsiasi buon testo di matematica; ma ciò che ora devi ricordare è che per convertire un angolo dai gradi in radianti, devi dividerlo per 57.2957805. Perciò se vuoi sapere il seno di un angolo di 40 gradi, digiterai:

```
PRINT SIN(40/57.2957805)
```

In modo analogo se avessimo una tangente e volessimo conoscere l'angolo corrispondente, la funzione ATN farà al caso nostro e darà una risposta in radianti, che moltiplicherai per 57.2957805 per trasformarla in gradi. Se per es. avessimo una tangente di 0.8391, useremmo:

```
PRINT ATN(.8391)*57.2957805
```

che dà come risultato 40 gradi circa.

Altre funzioni

Ci sono poche altre funzioni numeriche che non abbiamo spiegato e che sono elencate qui di seguito.

HEX$

La funzione HEX$ è l'opposto della funzione DEC. Essa trasforma un numero decimale in un numero esadecimale che può essere inserito in una variabile stringa.

```
PRINT HEX$(702)
```

dà il risultato 2BE.

FRE

La funzione FRE è utile perché ti dice quanta memoria hai usato per il tuo programma. È usata così:

```
PRINT FRE(0)
```

Ti dice esattamente quanti bytes di memoria ti rimangono. Per trasformare questo numero in kilobytes di memoria, devi dividere il numero di bytes per 1024. Il numero tra parentesi dopo la funzione FRE può essere un numero qualsiasi, e non influenza il risultato.

POS

Questa funzione ti dice da quale colonna partirà la prossima istruzione PRINT. Come la funzione FRE, ha dopo di sé un numero qualsiasi tra parentesi. Ecco un esempio:

```
Z=POS(0)
```

SPC

L'unica principale differenza tra SPC e TAB è che la funzione SPC lavora con la stampante, mentre TAB non lo fa. SPC è usata esattamente nello stesso modo di TAB e fa esattamente le stesse cose, così il comando:

```
PRINT SPC(5);"CIAO"
```

avrà lo stesso effetto di:

```
PRINT TAB(5);"CIAO"
```

che fa apparire il messaggio CIAO 5 spazi a sinistra sullo schermo.

CAPITOLO 11

LINGUAGGIO MACCHINA

PEEK e POKE

Ormai hai una conoscenza appropriata sul BASIC e presto incomincerai a pensare "se solo potessi far muovere l'astronave più velocemente" oppure "posso far scorrere lo schermo da entrambe le parti, in su e in giù?". A questo punto sei pronto per esplorare il mondo della programmazione in linguaggio macchina. Il tuo computer ha una agevolazione per renderlo più facile, che viene chiamata TEDMON. Ma prima di esaminarla, dobbiamo imparare come lavora la memoria del computer e come puoi prendervi (PEEK) e introdurvi (POKE) dei dati.

Il tuo computer ha due tipi di memoria, ROM (Read Only Memory = memoria a sola lettura) e RAM (Random Acces Memory = memoria ad accesso casuale). L'uso di ognuna di queste memorie è molto complesso, ma cercherò di dare una spiegazione semplificata.

Immagina che nel tuo computer ci siano migliaia di scatole di vetro, ognuna con un numero, in modo che tu possa sapere di che scatola si tratta (questo numero rappresenta un indirizzo di memoria) e che all'interno di ogni scatola ci sia un pezzo di carta con un numero da 0 a 255 (questo numero rappresenta il contenuto della memoria).

Alcune scatole hanno un coperchio e non possono essere aperte, altre invece non ce l'hanno. Le scatole con il coperchio rappresentano la ROM, quelle senza rappresentano la RAM.

Dato che le scatole sono fatte di vetro, possiamo guardare all'interno di ognuna di esse per vedere quale numero è scritto sul pezzo di carta. Possiamo anche prendere il pezzo di carta all'interno delle scatole aperte (RAM) e metterci un altro pezzo di carta con un numero diverso. Per quanto riguarda le scatole chiuse (ROM), anche se puoi leggere il contenuto della memoria, non puoi cambiarla.

Ci sono molti tipi differenti di ROM. C'è una unità centrale di elaborazione, la CPU (Central Processing Unit) dove, come suggerisce il nome, viene elaborata ogni cosa, o se preferisci è il "cervello" del computer. Il problema con la CPU è che capisce solo il suo linguaggio e non il BASIC. Per tradurre i comandi in BASIC che tu dai, in qualcosa che esso può comprendere, necessita di un altro tipo di ROM, l'interprete.

Avrai probabilmente osservato che quando spegni il tuo computer viene perso ogni programma immagazzinato nella memoria. Il computer comunque non perde mai il linguaggio BASIC perché la RAM necessita di elettricità per conservare i suoi contenuti, mentre la ROM è un tipo di memoria permanente. Un po' della RAM è usata dal computer come spazio di lavoro — tutto quello che è sullo schermo, ad es., viene immagazzinato nella RAM. Quindi non puoi usare tutta la RAM per il tuo programma.

Il comando PEEK ti permette di conoscere il contenuto della memoria, per vedere cosa è stato immagazzinato. Puoi usarlo sia per la RAM che per la ROM.

Il comando POKE ti permette di cambiare il contenuto della memoria, cambiando solo il numero sul pezzo di carta nella scatola di vetro. Puoi usarlo solo con la RAM perché la ROM non può essere cambiata. Se lo usi in un indirizzo ROM non avrà alcun effetto e non verrà fatto alcun danno.

PEEK e POKE sono utili per alterare ciò che è sullo schermo. Se pulisci lo schermo e digiti:

```
POKE 3072,1
```

vedrai apparire nell'angolo in alto a sinistra una lettera A. Il numero 3072 è l'indirizzo della memoria dell'angolo in alto a sinistra dello schermo. Il numero 1 è il codice della lettera A (vedi Appendice D). Puoi cambiare il numero 3072 con qualsiasi altro numero tra 3072 e 4072 e vedrai che puoi far apparire la A in qualunque punto sullo schermo. Puoi anche far apparire caratteri differenti cambiando l'1 con un numero qualsiasi tra 0 e 255.

Se digiti

```
PRINT PEEK(3072)
```

vedrai apparire sullo schermo il numero 1 (a condizione che la A sia ancora nell'angolo sullo schermo) perché abbiamo detto al computer di guardare nella locazione di memoria 3072 e mostrarne il contenuto sullo schermo.

Se provi a guardare (PEEK) in una differente locazione di memoria dallo 0 al 65535 vedrai il contenuto di quelle locazioni di memoria (puoi anche cercare di inserirci (POKE) qualcosa, ma non sorprenderti se succedono strane cose).

Memoria del colore

Sai già come fare per mettere dei caratteri sullo schermo usando il comando POKE, ma è possibile cambiare il colore di qualsiasi carattere che sia sullo schermo e farlo lampeggiare, usando il POKE. Digita questo:

```
POKE 3072,1:POKE2048,128
```

Vedrai apparire nell'angolo in alto a sinistra sullo schermo una lettera A lampeggiante.

Per cambiare il colore del carattere sullo schermo e per farlo lampeggiare, devi fare tre cose. Prima devi decidere di quale colore vuoi che sia il carattere; il numero del colore sarà da 1 a 16 come con il comando COLOR. Devi poi decidere quale valore di luminosità desideri; esso va da 0 a 7 e deve essere moltiplicato per 16 e aggiunto al numero del colore. Se vuoi che il carattere lampeggi, devi aggiungere 128 al risultato. Il valore finale che otterrai deve essere diminuito di 1 e inserito (POKE) nella locazione di memoria pertinente.

La memoria del colore va da 2048 a 3048. Il modo più facile per scoprire dove devi inserire (POKE) il colore è di sottrarre 1024 dalla locazione di memoria del carattere da colorare. Se è stata inserita una lettera A nella locazione di memoria 3116, che è la terza posizione carattere orizzontale e la seconda verticale, allora devi sottrarre 1024 da 3116, ottenendo 2092 e inserire il codice di colore che vuoi nella locazione di memoria 2092.

Ecco un programma che inserisce (POKE) cerchietti colorati a caso in una parte qualunque dello schermo:

```
10 COLOR 4,2,7:COLOR 0,2,7:SCNCLR
20 X=INT(RND(0)*999)+1
30 C=INT(RND(0)*15)+1
40 I=INT(RND(0)*6)
50 C=C+I*16-1
60 POKE X+2047,C:POKEX+3071,81
70 GOTO 20
```

Il programma funziona così:

Linea 20: Sceglie un numero a caso tra 0 e 1, lo moltiplica per 999 e lo arrotonda per difetto prima di aggiungere 1 al risultato e assegnare il risultato alla variabile X.

Linea 30: Sceglie un numero a caso tra 0 e 1, lo moltiplica per 15 e lo arrotonda per difetto prima di aggiungere 1 al risultato e assegnare il risultato alla variabile C.

Linea 40: Sceglie un numero a caso tra 0 e 1, lo moltiplica per 6 e lo arrotonda per difetto prima di assegnare il risultato alla variabile 1.

Linea 50: Moltiplica il valore della variabile I per 16 e aggiunge il risultato alla variabile C.

Linea 60: Immagazzina il valore della variabile C nella locazione di memoria 2047 più il valore della variabile X, poi immagazzina il valore 18 nella locazione di memoria 3071 più il valore della variabile X.

Introduzione a TEDMON

Il linguaggio macchina è il linguaggio della CPU del tuo computer. La programmazione in linguaggio macchina permette di eseguire alcuni programmi più velocemente rispetto al BASIC, perché la CPU non ha bisogno di un interprete che traduce tutti i comandi.

Questo capitolo non vuole essere una introduzione alla programmazione in linguaggio macchina, dato che sarebbe possibile scrivere un intero libro su questo argomento, ma una introduzione a TEDMON, un monitor in linguaggio macchina interno al tuo computer, che ti permette di scrivere programmi in linguaggio macchina. TEDMON è pronto per l'uso; per richiamarlo devi semplicemente digitare MONITOR. Il computer mostrerà sullo schermo (i numeri potrebbero variare):

```
MONITOR   (è quello che hai digitato)
MONITOR
PC    SR AC XR YR SP
;FFFF 00 FF FF FF F9
```

TEDMON è pronto ed i numeri che appaiono sullo schermo, i registri, ti danno delle informazioni sul computer. Vedremo più avanti cosa dicono in realtà questi numeri, e per ora diamo uno sguardo a cosa può fare TEDMON.

Mostrare il contenuto della memoria

Prova a digitare:

```
M 8188
```

Vedrai allora apparire questo sullo schermo (i caratteri che seguono i due punti su ogni linea appariranno in negativo).

```
>8188 02 A9 5A 4C 94 04 45 4E:.>ZL..EN
>8190 C4 46 4F D2 4E 45 58 D4:DFORNEXT
>8198 44 41 54 C1 49 4E 50 55:DATAINPU
>81A0 54 A3 49 4E 50 55 D4 44:T#INPUTD
>81A8 49 CD 52 45 41 C4 4C 45:IMREADLE
>81B0 D4 47 4F 54 CF 52 55 CE:TGOTORUN
>81B8 49 C6 52 45 53 54 4F 52:IFRESTOR
>81C0 C5 47 4F 53 55 C2 52 45:EGOSUBRE
>81C8 54 55 52 CE 52 45 CD 53:TURNREMS
>81D0 54 4F D0 4F CE 57 41 49:TOPONWAI
>81D8 D4 4C 4F 41 C4 53 41 56:TLOADSAV
>81E0 C5 56 45 52 49 46 D9 44:EVERIFYD
```

Le lettere sulla parte destra dello schermo ti sembreranno familiari e più tardi chiariremo il perché.

Il segno 'maggiore di' è all'inizio di ogni linea ed è lì per permetterti di alterare gli otto numeri di due cifre. L'uso completo del comando verrà spiegato più avanti.

Il numero di quattro cifre immediatamente dopo il segno 'maggiore di' è un indirizzo di memoria esadecimale. Ci sono dieci diverse cifre nella numerazione decimale. L'esadecimale, o in base 16, ha 16 differenti cifre, da 0 a 9 e da A a F dove la A rappresenta 10, B rappresenta 11 e così via. Ci sono due funzioni in BASIC, HEX$ e DEC che convertono i numeri da decimali in esadecimali, così se hai bisogno di convertire da una base all'altra, puoi ritornare al BASIC e usare queste funzioni per fare la tua conversione.

8188 è 33160 in decimali ed è una locazione dell'interprete ROM. Gli otto numeri esadecimali dopo l'indirizzo sono i contenuti della locazione di memoria 8188 e le 7 seguenti. Ciò significa che il contenuto della locazione di memoria 8188 è 02, il contenuto della locazione di memoria 8189 è A9 e così via.

Alla fine di ogni linea c'è una serie di caratteri in negativo. Gli otto numeri esadecimali a due cifre sono i codici ASCII di questi caratteri. Dove un particolare carattere non è stampabile viene mostrato un punto.

L'area particolare di memoria che stai guardando è la Tabella delle parole riservate dell'interprete. Se vuoi continuare ad esaminare la ROM dovrai digitare M e poi RETURN ogni volta che vuoi vedere la sezione successiva di memoria. In alternativa puoi digitare:

```
M 8188 8382
```

per far apparire tutta la tabella delle parole riservate. Le scritte scorreranno abbastanza facilmente ma possono essere rallentate con il tasto Commodore.

Il primo numero dopo il comando M è la locazione di partenza e il secondo numero è la locazione finale della parte di memoria da mostrare. Ciò significa che il precedente comando M dice a TEDMON di far apparire i contenuti di tutte le locazioni di memoria tra 8188 e 8382.

È stato menzionato in precedenza che il comando > (maggiore di) ti permette di alterare il contenuto delle locazioni di memoria; questo segno viene mostrato automaticamente all'inizio di ogni linea, perciò puoi muovere il cursore sul numero che vuoi modificare e cambiarlo. Se provi a muovere il cursore su un numero dell'elenco che hai appena modificato, e poi premi il RETURN, vedrai il numero che hai cambiato, ritornare al suo valore originale. Questo succede perché i numeri sullo schermo sono letti dalla ROM e come ben sai, non puoi alterare i contenuti della ROM. Comunque se digiti:

```
M 3000
```

puoi modificare le locazioni di memoria che desideri, perché la memoria che ora è stata mostrata, fa parte della RAM.

Non hai bisogno di mostrare un'area della memoria perché ne cambi i contenuti. Invece puoi digitare un comando simile a questo:

```
>324A 3A BD F4
```

Questo comando dice al computer di immagazzinare il numero 3A nella locazione di memoria 324A, il numero BD nella locazione di memoria 324B e il numero F4 nella locazione di memoria 324C.

Puoi cambiare da 1 a 8 locazioni per volta usando questo comando, così se vuoi cambiare le 8 locazioni di memoria da 2BC2 in avanti, puoi digitare:

```
>2BC2 23 B4 6A DA 9F E2 FC 14
```

Uscita da TEDMON

È molto facile lasciare TEDMON e tornare al BASIC; tutto quello che devi fare è digitare:

```
X
```

(naturalmente premere il RETURN) e tornerai al BASIC.

Riempire un'area della memoria

È possibile anche riempire un'area di memoria con un certo valore. Per es. se vuoi riempire le locazioni di memoria da 2400 a 2A00, con il valore A3 digiterai:

```
F 2400 2A00 A3
```

Se ora digiti:

```
M 2400 2A00
```

vedrai che ora tutte queste locazioni di memoria contengono il valore A3.

Ricerca di numeri e stringhe

Un'altra possibilità è la ricerca (HUNT). Se digiti:

```
H 7000 9000 C0
```

il computer cercherà tutte le locazioni di memoria dal 7000 al 9000 che contengono il numero C0. I risultati della ricerca verranno mostrati e saranno

```
80AB 83E4 842D 87AB 89A1 8B10 8BDD 8C12 8EB3 8EE1
```

Ognuno di questi numeri è l'indirizzo di una locazione di memoria che contiene il numero C0. Se controlli queste locazioni di memoria, vedrai che è esatto.
Puoi cercare anche una stringa di caratteri. Se digiti:

```
H 8000 9000 'COMMODORE
```

allora avrai:

```
80CF
```

Se ora digiti:

```
M 80CF
```

vedrai che la parola 'COMMODORE' è, infatti, immagazzinata in quell'area di memoria, con la 'C' di 'COMMODORE' immagazzinata nella locazione di memoria 80CF.
Se guardi il comando H (ricerca) vedrai che facciamo precedere alla stringa di caratteri da cercare un apostrofo. Esso dice al computer che vogliamo cercare una stringa di caratteri e non un numero.

Trasferimento di blocchi di memoria

Un comando molto utile è il comando T (trasferimento). Se digiti:

```
T 0C00 0FFF 0BD8
```

vedrai muoversi in su di una linea tutto ciò che è sullo schermo. In realtà abbiamo detto al computer di trasferire i contenuti delle locazioni di memoria dalla 0C00 alla 0FFF (che è la memoria dello schermo) nelle locazioni di memoria 0BD8 in avanti. SE 0BD8 è 40 locazioni di memoria prima di 0C00, questo trasferimento ha l'effetto di far scivolare lo schermo in su di una linea.
Come puoi vedere il comando di trasferimento necessita 3 numeri dopo di sé. Il primo è l'indirizzo di partenza del blocco di memoria che vuoi trasferire e il secondo è l'indirizzo finale di quel blocco. L'ultimo numero è l'indirizzo di partenza nel quale verrà trasferito il blocco di memoria.

Programmi scritti in linguaggio macchina

TEDMON ti permette anche di scrivere programmi in linguaggio macchina. Prova digitando questo breve programma:

```
A 2000 LDA #$01
A 2002 STA $0C00
A 2005 LDA #$80
A 2002 STA $0800
A 200A BRK
```

Nel momento in cui digiti ogni linea del programma, il computer muoverà ciò che hai digitato più avanti sullo schermo e farà apparire dei numeri all'inizio. Automaticamente apparirà un'altra A e un numero all'inizio della linea seguente. Quando arrivi alla fine del programma devi solo premere il tasto RETURN. Apparirà una cosa del genere:

```
A 2000   A9 01      LDA #$01
A 2002   8D 00 0C   STA $0C00
A 2005 - A9 80      LDA #$80
A 2002   8D 00 08   STA $0800
A 200A   00         BRK
A 200B
```

La lettera A all'inizio di ogni linea è il comando ASSEMBLE e dice al computer che ciò che stai digitando è un comando in linguaggio macchina. Il numero successivo alla A è l'indirizzo di memoria nel quale il comando in linguaggio macchina verrà immagazzinato. Devi solo dire al computer l'indirizzo di memoria dal quale vuoi partire per inserire il codice di L.M. (linguaggio macchina) e il computer farà il resto da solo.

I numeri che il computer aggiunge prima del comando in L.M. quando premi il RETURN, sono gli equivalenti in L.M. di questi comandi. Ogni comando in L.M ha un numero di codice: ad es. il comando LDA = S01 ha il numero di codice A9. Questo numero di codice cambia a seconda di come il comando è stato usato, come vedrai in seguito.

Uno o due numeri possono seguire il numero di codice; essi seguono sia il comando in L.M. sia il comando nella linea del nostro programma. LDA = S01 è stato cambiato con A901, e lo 01 è rimasto invariato. Quando però gli indirizzi di memoria sono stati usati come avviene nella seconda linea, le cose cambiano leggermente. STA S0C00 diventa 8D 00 0 C e significa che le parti 0C e 00 del numero sono invertite; questo perché il computer deve avere il byte basso di un numero (00) prima del byte alto (0C).

Infine arriviamo ai comandi. Non intendo parlare della programmazione in L.M. nei particolari, ma solo dare una breve spiegazione di cosa fa ogni comando nel programma.

LDA = S01: Carica l'Accomulatore con il valore esadecimale 01. L'Accomulatore può essere paragonato ad una variabile.
STA S0C00: Immagazzina il valore dell'Accomulatore nella locazione di memoria 0C00 (esadecimale). Questo è l'indirizzo dell'angolo in alto a sinistra sullo schermo.
LDA = S80: Carica l'Accomulatore con il valore esadecimale 80 (128 decimale).
STA S0800: Immagazzina l'Accomulatore nell'indirizzo di memoria 0800. Questa è la locazione di memoria del colore per l'angolo in alto a sinistra sullo schermo.
BRK: Finisce il programma e il controllo ritorna a TEDMON.

Esecuzione di programmi in linguaggio macchina

Puoi far eseguire questo programma digitando:

```
G 2000
```

Appena hai premuto il tasto RETURN apparirà nell'angolo in alto a sinistra sullo schermo una lettera A nera lampeggiante.

Il comando *GO* dice al computer di partire per eseguire un programma in L.M.. Devi dire al computer dove si trova nella memoria la partenza del programma, come abbiamo fatto nell'esempio precedente, dicendo al computer di incominciare ad eseguire la procedura in L.M. partendo dalla locazione di memoria 2000 (esadecimale).

Disassemblare programmi in linguaggio macchina

TEDMON può sia assemblare un programma in L.M., cioè convertirlo da una serie di comandi in una serie di numeri, sia disassemblare un programma in L.M., cioè convertire tutti i numeri nei comandi che essi rappresentano. Per farlo usiamo il comando D (disassemblare). Prova questo esempio:

```
D 9000
```

Vedrai allora apparire sullo schermo:

```
.9000  F0 3C     BEQ $903E
.9002  C9 FB     CMP #$FB
.9004  D0 03     BNE $9009
.9006  4C F7 AE  JMP $AEF7
.900B  C9 A3     CMP #$A3
.900D  F0 50     BEQ $905F
.900F  C9 A6     CMP #$A3
.9011  18        CLC
.9012  F0 4B     BEQ $905F
.9014  C9 2C     CMP #$2C
```

Questo disassemblato è nel formato in cui il computer converte i tuoi programmi in L.M. quando li assembla, come potrai vedere se li confronti, sebbene i comandi e i numeri siano differenti. Se l'area di memoria che hai disassemblato è in RAM allora puoi alterare i contenuti della memoria muovendo il cursore sopra il comando che vuoi alterare e poi modificarlo. Se digiti:

```
D 2000 200A
```

vedrai il tuo programma in L.M. sullo schermo. Muovi il cursore verso la prima linea e fai le modifiche:

```
.2000 A9 01 LDA #$02
```

Appena premi il tasto RETURN questa linea cambierà in:

```
A 2000 A9 02 LDA #$02
```

Il punto all'inizio della linea cambierà con una lettera A (che sta per assemblaggio) e il numero 01 cambierà con 02. Anche il punto all'inizio della linea seguente cambierà con una lettera A e il cursore sarà sopra la prima cifra del numero 2002. Il comando di disassemblaggio ha lo stesso formato del comando per la lettura della memoria, così se digiti:

```
D 4000 4020
```

il computer disassemblerà il contenuto della locazione di memoria 2000 in 4020 (esadecimale).

Confronto di blocchi di memoria

Il comando COMPARE confronta un blocco di memoria con un altro e ti dice dove ci sono delle differenze. Se digiti:

```
C 1000 2000 4000
```

allora lo schermo si riempie con locazioni di memoria, perché molte di quelle dal 1000 al 2000 sono diverse da quelle dal 4000 al 5000; abbiamo perciò detto al computer di paragonare le locazioni di memoria dal 1000 al 2000 con quelle dal 4000 al 5000. Dobbiamo dare al computer solo l'indirizzo di partenza del secondo blocco di memoria, perché ne trova da solo la fine.

Il computer mostrerà un elenco di locazioni di memoria che differiscono tra loro, in modo che se due blocchi di memoria sono uguali il computer non mostrerà niente. Comunque se i blocchi di memoria sono totalmente differenti, lo schermo si riempirà rapidamente con indirizzi di memoria.

Salvare programmi in linguaggio macchina

Se hai scritto un programma in L.M. probabilmente vorrai registrarlo su nastro o su dischetto. TEDMON lo fa per te, basta che tu gli dica il nome del programma e le locazioni di inizio e di fine della memoria da registrare. Se vuoi, ad es., salvare un programma chiamato LEFT SCROLL su nastro e questo programma è stato immagazzinato nelle locazioni di memoria dal 3000 al 3040, digiterai:

```
S "LEFT SCROLL",1,3000,3041
```

Osserverai che dobbiamo aggiungere 1 all'indirizzo finale di memoria, perché il computer salva tutta la memoria tra la locazione di partenza fino alla locazione finale, senza includerla. Se vuoi registrare il tuo programma su dischetto dovrai cambiare il numero del dispositivo (l'1 nel comando SAVE precedente) con un 8, così:

```
S "LEFT SCROLL",8,3000,3041
```

Caricare programmi in linguaggio macchina

È facile caricare dei programmi in L.M., basta dire a TEDMON il nome del programma che desideri caricare e se vuoi caricarlo da nastro o da dischetto. Così se vuoi caricare un programma in L.M. chiamato RANA da nastro, digiterai:

```
L "RANA",1
```

e per caricare lo stesso programma da dischetto, digiterai:

```
L "RANA",8
```

L'1 dopo il nome dice a TEDMON che vuoi caricarlo da nastro, mentre l'8 dice di caricarlo da dischetto.

Verificare programmi in linguaggio macchina

È possibile verificare un programma in L.M. esattamente nello stesso modo in cui si verifica un programma in BASIC. Per farlo devi dire a TEDMON il nome del programma che deve essere verificato e se è registrato su nastro o su dischetto, esattamente come fai quando carichi un programma in L.M., così:

```
V "COMPOSITORE",1
```

ciò dice al computer di verificare il programma su nastro con il nome COMPOSITORE, con il programma in L.M. che è attualmente in memoria. Se i due programmi sono uguali, il cursore lampeggiante ritornerà, mentre se sono differenti apparirà il messaggio VERIFY ERROR.

Per verificare un programma su dischetto con il nome COMPOSITORE, digiterai:

```
V "COMPOSITORE",8
```

I comandi di salvataggio, caricamento e verifica come i loro equivalenti in BASIC, fanno diventare lo schermo bianco usando il registratore, mentre appaiono i soliti messaggi.

I registri

Parliamo infine dei registri. Sono le lettere e i numeri che appaiono quando entri per la prima volta in TEDMON e ti danno delle informazioni sul computer.

Il primo registro si chiama PC cioè *Program Counter* (Contatore di Programma). Esso indica sempre la parte del programma in L.M. che il computer sta eseguendo. Per es. se il computer sta eseguendo un programma in L.M. dalle locazioni di memoria 3A42 in avanti allora il contatore del programma conterrà l'indirizzo di memoria 3A42, 3A43, 3A44 e così via.

Il secondo registro è lo *Status Register* (Registro di Stato) e contiene informazioni sulle operazioni che sono appena state eseguite.

Il terzo registro è l'*Accomulator* (Accomulatore); potrebbe sembrare simile ad una variabile, ma è usato in modo leggermente diverso. I registri X e Y sono simili all'Accomulatore, anche se ognuno può fare cose che gli altri non possono fare.

L'ultimo registro è lo *Stack Pointer* (Stack = catasta, pila). È un'area di memoria nella quale vengono immagazzinati dal computer dei numeri, in modo simile ad una pila di libri. Puoi aggiungere dei libri alla pila e puoi anche toglierli, ma puoi aggiungerli solo in cima e non a metà, come puoi toglierli solo dalla cima, uno alla volta, Questo significa che l'ultimo libro aggiunto alla pila è il primo che deve essere tolto e, nello stesso modo, il primo numero aggiunto al gruppo è l'ultimo ad essere rimosso. Lo Stack è lungo 256 bytes, cioè può contenere 256 bytes. Lo stack Pointer indica la prima locazione di memoria libera nello Stack.

Puoi esaminare i registri in ogni momento digitando:

```
R
```

I registri e i loro contenuti appariranno sullo schermo. Se lo desideri puoi alterare i contenuti dei registri usando il comando punto e virgola (;). Questo comando viene mostrato all'inizio dei contenuti del registro, perciò tutto quello che devi

fare è muovere il cursore sul contenuto del registro che vuoi cambiare e modificarlo. Per es. se i registri avessero questi valori:

```
PC    SR AC XR YR SP
;BCB1 00 AF BF 28 F9
```

e volessi cambiare i contenuti del registro X con 2A, dovresti muovere il cursore sopra la B di BF, così:

```
PC    SR AC XR YR SP
;BCB1 00 AF BF 28 F9
```

poi digiti 2A e premi il RETURN. I contenuti del registro X cambieranno con 2A.

Il comando SYS

Se vuoi eseguire il tuo programma in L.M. dal BASIC, hai bisogno di aggiungere una istruzione RTS alla fine della procedura. Per es. se disassembli la tua breve procedura in L.M. e fai questo cambiamento:

```
A 200A RTS
```

poi esci da TEDMON e digiti SYS DEC ("2000"), il tuo programma in L.M. verrà eseguito e la A lampeggiante apparirà nell'angolo in alto a sinistra sullo schermo.

Il comando SYS dovrebbe essere seguito dall'indirizzo di partenza del programma in L.M. Nell'esempio precedente abbiamo incluso anche la funzione DEC che convertirà il numero 2000 in un decimale prima che il comando SYS esegua il programma in L.M., che parte da quella locazione di memoria.

La funzione USR

Un modo alternativo per eseguire un programma in L.M. dal BASIC si ha usando la funzione USR. Essa viene usata per passare un numero o una stringa di caratteri ad un programma in L.M. e farlo eseguire. Comunque devi immagazzinare l'indirizzo di partenza del programma in L.M. nelle locazioni di memoria 1281 e 1282. Per es. per eseguire il nostro programma in L.M. usando il comando USR dobbiamo utilizzare questa procedura:

```
PRINT DEC("2000")
8192
```

```
PRINT 8192/256
32

READY.
POKE 1281,0:POKE 1282,32
READY.
X=USR(0)
```

Il programma verrà così eseguito:

In realtà abbiamo convertito il numero 2000 esadecimale in decimale, dividendolo per 256. Poi inseriamo il byte basso nella locazione di memoria 1281 e il byte alto, che è 32, nella locazione di memoria 1282. Affinché USR sia una funzione, deve essere usata nel formato:

variabile = USR (valore)

Il valore tra parentesi è quello che vuoi far passare nel programma in L.M. ma dato che non abbiamo bisogno di far passare alcun valore particolare nel nostro programma, non ha importanza quale valore usiamo. La variabile che viene usata avrà un valore immagazzinato in essa nel momento in cui finisce la routine del L.M., perciò non usare una variabile che è già stata utilizzata per qualcos'altro.

Ora dovresti avere una discreta conoscenza sull'uso di TEDMON e quando deciderai di imparare la programmazione in L.M., sarebbe una buona idea rileggerti questo capitolo, per imparare bene ad usare le tante facilitazioni di TEDMON.

CAPITOLO 12

PERIFERICHE

Uso dell'unità dischi (disk drive)

Il disk drive è estremamente utile perché ti permette di salvare e caricare i tuoi programmi più velocemente di quanto non si possa fare con un registratore. Potresti usare sia il C1541, il C1542 (l'unica diversità tra questi due modelli è il colore) o lo SFS 481, disk drive veloce, tutti distribuiti dalla Commodore.

Il C1541 o il C1542 deve essere collegato con la presa contrassegnata da SERIAL, sulla parte posteriore del tuo computer e l'altra estremità del cavo deve essere inserita in uno dei due connettori DIN sul retro del disk drive. Lo SFS 481 deve essere collegato con la presa contrassegnata USER PORT sul retro del tuo computer.

Quando il disk drive è collegato avrai bisogno di qualche dischetto da 5 pollici e un quarto, singola faccia, doppia densità, per salvare i tuoi programmi. La maggior parte dei negozi di computer, e anche qualche rivenditore all'ingrosso, vende questi dischetti.

Precauzioni

Con i dischetti devi avere alcune precauzioni. Il dischetto si trova dentro un involucro quadrato di plastica che serve per proteggerlo. C'è una finestrella in questo involucro attraverso la quale puoi vedere il dischetto. Non devi toccarlo attraverso questa fenditura. Il dischetto è fatto di materiale simile a quello usato per le cassette e come esse, anche il dischetto non deve essere messo vicino ad un magnete (includendo anche la televisione o la parte superiore del disk drive) né deve essere esposto al sole. I dischetti devono essere tenuti in un ambiente ad una temperatura tra i 10 e i 51 gradi centigradi, o dai 50 ai 125 gradi Fahrenheit e si deve sempre fare attenzione a non piegarli.

Etichetta di protezione dalla scrittura

Prima di salvare qualsiasi cosa sul dischetto è necessario formattarlo. Per farlo devi prima assicurarti che la tacca sulla parte sinistra del dischetto non sia coperta da una etichetta. Infatti se è coperta non si può salvare niente sul dischetto o fare qualcosa per alterarne il contenuto.

Quando ti sei assicurato che la tacca sia libera, puoi inserire il dischetto nel disk drive. Apri lo sportello del disk drive spingendolo verso l'interno e inserisci il dischetto nel drive con la tacca di protezione sulla sinistra e la fenditura che ti permette di vedere il dischetto di fronte al disk drive. Poi chiudilo spingendo verso il basso. Se hai fatto tutto correttamente, la porta scatterà a posto.

Inizializzazione dei dischetti

Ora che hai inserito correttamente il dischetto nel disk drive, avrai bisogno di inizializzarlo. Il computer suppone che il dischetto sia diviso in tracce e settori. Ci dovrebbero essere 35 tracce di 17/21 settori. Ogni settore può contenere 256 bytes. Per dare al dischetto questo formato dobbiamo usare il comando HEADER. Assicurati di avere un dischetto nel drive e poi digita:

```
HEADER "DISCO 1",I01,D0,U8
```

Quando l'hai fatto il computer ti chiederà:

`ARE YOU SURE?` (sei sicuro?)

a cui tu rispondi Y (Yes = Sì) se sei sicuro che vuoi inizializzare questo dischetto. Allora il drive partirà e si accenderà la spia rossa. Ci vuole abbastanza tempo per inizializzare un dischetto, perciò ti conviene sederti e aspettare fino a quando il drive si ferma.

Se la spia rossa continua a lampeggiare quando il drive si è fermato, vuol dire che il dischetto non è stato inizializzato. Per sapere cosa c'è di sbagliato devi digitare:

```
PRINT DS$
```

Il computer ti darà un messaggio d'errore e la spia rossa smetterà di lampeggiare. Dovrai allora controllare che il dischetto sia inserito correttamente e che l'etichetta sulla tacca sia stata tolta. Prova ad inizializzare il dischetto due o tre volte e se proprio non ci riesci, usa un altro dischetto. Se il secondo viene inizializzato correttamente significa che il primo era difettoso. Se dopo parecchi tentativi con altrettanti dischetti non riesci ancora ad inizializzarlo, sarebbe meglio portare il tuo disk drive dal rivenditore e spiegare qual è il problema.

In realtà digitando il comando HEADER hai detto al computer di dividere il dischetto nel numero corretto di tracce e settori e dargli il nome di DISCO 1. Inoltre hai dato al dischetto un numero di identità ,01, che il computer mette su ogni settore del dischetto. Il D0 dice al computer che vuoi inizializzare il dischetto nel drive 0 (il primo drive) e viene aggiunto U8 perché 8 è il numero del disposirivo del disk drive.

La DIRECTORY del dischetto

Il comando HEADER non divide solo il dischetto nel formato giusto, ma inserisce anche un indice o directory, sul dischetto. Puoi trovare quali programmi sono stati immagazzinati sul dischetto digitando semplicemente:

```
DIRECTORY
```

Se al momento il dischetto è vuoto, riceverai il messaggio:

```
0 "DISCO 1            "01 2A
664 BLOCKS FREE
```

DISCO 1 è il nome del dischetto e 01 è il numero di identità che hai dato al dischetto. Il 2A alla fine è un numero d'identità che il computer ha dato al dischetto.

Puoi anche far apparire tutti i file che incominciano con un certo carattere. Per es. se vuoi conoscere tutti i file sul dischetto che incominciano con i caratteri PROG, allora digiterai:

```
DIRECTORY "PROG*"
```

Come puoi vedere la parola PROG è chiusa tra virgolette ed è seguita da una stella. Essa dice al computer che vuoi vedere tutti i file che incominciano con i caratteri PROG.

Puoi rallentare la velocità alla quale la directory fa apparire i file, tenendo premuto il tasto COMMODORE. Lo scorrimento dello schermo può essere fermato completamente premendo CONTROL e S e fatto ripartire premendo un qualsiasi altro tasto.

Salvare un programma

Ora sei pronto per salvare un programma su dischetto. La cosa più semplice da fare è di caricare uno dei tuoi programmi dal nastro e poi digitare:

```
DSAVE "NOME PROGRAMMA"
```

dove "nome programma" è il nome che vuoi dare al tuo programma. Può essere una combinazione di lettere e numeri, basta che inizi con una lettera e non sia più lungo di 16 caratteri. Il comando DSAVE agisce come il comando SAVE, solo che salva il programma su dischetto anziché su nastro; inoltre anche lo schermo rimane acceso nel momento in cui un programma viene salvato su dischetto.

Controllo del programma

Quando appare il messaggio READY puoi controllare se il tuo programma è stato salvato correttamente. Per farlo devi digitare:

```
VERIFY "NOME PROGRAMMA",8
```

Ancora una volta il nome del programma è quello del programma che hai appena salvato. L'8 aggiunto alla fine del comando VERIFY (che hai già usato per controllare i programmi su nastro) dice al computer che vuoi verificare il programma sul dischetto con il nome del programma che è attualmente in memoria. Il comando VERIFY funziona nello stesso modo sia con i dischetti che con le cassette, eccetto per il fatto che lo schermo non diventa bianco quando avviene la verifica di un programma su dischetto.

Se vuoi puoi digitare DIRECTORY e vedere se il nome del programma che hai appena salvato, è stato aggiunto nell'indice. Il numero prima del nome del programma è il numero di blocchi occupati dal programma (ogni blocco è composto da 256 bytes, così se moltiplichi il numero di blocchi occupati per 256 otterrai esattamente quanti bytes occupa il tuo programma).

Caricamento del programma

Ora che sai che il tuo programma è stato salvato in modo corretto sul dischetto, puoi caricarlo di nuovo. Digita NEW per cancellare il tuo programma dalla memoria del computer e poi digita:

```
DLOAD "NOME PROGRAMMA"
```

In un tempo sorprendentemente breve il tuo programma verrà caricato e sarà pronto per essere eseguito, listato o per fare ciò che desideri.

Non devi digitare l'intero nome del programma per caricarlo. Per es. se digiti:

```
DLOAD "RANA*"
```

allora il computer caricherà il primo programma che trova sul dischetto e che inizia con i caratteri RANA. In alternativa, se sai che il programma che desideri è il primo sul dischetto, potresti digitare:

```
DLOAD "*"
```

Cambiare il nome del programma

È molto facile cambiare il nome del programma su dischetto e spesso può essere utile. Devi solo usare il comando RENAME e dire al computer il vecchio nome del programma e il nuovo nome che vuoi dargli. Per es. se hai un programma sul dischetto che si chiama INVADERS e vuoi cambiarlo con COSMIC digiterai:

```
RENAME "INVADERS" TO "COSMIC"
```

Puoi cambiare il nome a qualsiasi cosa sul dischetto usando questo comando e impiegherai solo pochi secondi perché il computer deve solo aggiornare la directory.

Fare una copia in più del programma

Spesso è utile fare una copia in più del programma sul dischetto, per averne una di riserva se per caso cancelli l'originale. Per fare questa copia in più devi usare il comando COPY. Per es. se vuoi fare la copia di un programma chiamato OROLOGIO digiterai:

```
COPY "OROLOGIO" TO "OROLOGIO 2"
```

Osserverai che il nome della seconda copia è differente dall'originale. Ciò succede perché non puoi avere due programmi con lo stesso nome su uno stesso dischetto.

Per copiare un programma ci vuole un po' di tempo e più lungo è il programma, più tempo impieghi per copiarlo. Se hai due disk drives potresti copiare un programma da un dischetto ad un altro. Così se vuoi copiare il programma OROLOGIO dal drive 0 al drive 1 digiterai:

```
COPY D0,"OROLOGIO" TO D1,"OROLOGIO"
```

In questo caso puoi dare alla seconda copia lo stesso nome dell'originale perché i due programmi sono su due dischetti separati.

Il comando COPY può essere usato anche per copiare tutto il dischetto su un altro, se hai due disk drives. Basta digitare:

```
COPY D0 TO D1
```

Cancellare un programma dal dischetto

Qualche volta ti succederà di voler togliere un programma dal dischetto. Per farlo usi il comando SCRATCH. Per es. se vuoi rimuovere un programma chiamato AVVENTURA digiterai:

```
SCRATCH "AVVENTURA"
```

Il computer ti chiederà

```
ARE YOU SURE?
```

a cui risponderai premendo Y o N.

Salvare di nuovo un programma

Se vuoi salvare una versione aggiornata di un programma, al posto dell'originale, devi aggiungere un simbolo @ prima del nome del programma nel comando DSAVE. Per es. se hai aggiornato un programma chiamato RANA e vuoi salvare la nuova versione al posto della vecchia, digiterai:

```
DSAVE "@:RANA*"
```

e il computer salverà il tuo programma sovrapponendolo al vecchio programma RANA.

Riordino del dischetto

Se hai usato un dischetto per tanto tempo, avrai certamente salvato e cancellato dei programmi e ciò significa che ci saranno anche degli spazi vuoti. Inoltre se hai usato dei file (di cui si parlerà più avanti), potrebbero essercene alcuni chiusi in modo scorretto o che sono in disordine sul dischetto. Per riordinare il tutto dovrai usare il comando COLLECT, in questo modo:

```
COLLECT
```

Con questo unico comando il disk drive partirà e dopo una breve pausa il dischetto sarà stato riordinato, lasciandoti più spazio per immagazzinare altri programmi.

Copia di un dischetto (BACKUP)

Il comando BACKUP può essere usato solo se hai più di un disk drive, perché serve per copiare il contenuto di un intero dischetto su un altro dischetto. Per es. per copiare tutto quello che c'è dal dischetto nel drive 0, sul dischetto nel drive 1, dovrai digitare:

```
BACKUP D0 TO D1
```

Il computer procederà a copiare il contenuto da un dischetto all'altro, cancellando tutto ciò che era sul secondo dischetto. Il secondo dischetto può non essere inizializzato e il comando BACKUP ne tiene conto.

Nota: Tutti i comandi per il dischetto possono essere usati nello stesso modo, sia che tu abbia un solo drive, due, tre o perfino quattro. Per usare un comando qualsiasi con un disk drive qualunque, diverso dal drive 0, devi aggiungere ,D e poi il numero del drive che desideri usare, alla fine di qualunque comando. Per es. se vuoi guardare la DIRECTORY del secondo disk drive digiterai:

```
DIRECTORY D1
```

e per caricare un programma dal terzo disk drive digiterai:

```
DLOAD "NOME PROGRAMMA",D2
```

Uso della stampante

La stampante è un dispositivo estremamente utile che ti permette di fare delle copie dello schermo o di stampare dei listati dei tuoi programmi sulla carta, che naturalmente puoi conservare nel caso venga persa la copia originale del programma.

La maggior parte delle stampanti per il tuo computer sono fatte dalla Commodore (fai attenzione a quelle che non lo sono, ad alcune che non stampano i caratteri grafici e ad altre che hanno bisogno di cavi e interfacce in più per funzionare). Il cavo della stampante viene inserito nel connettore contrassegnato da SERIAL. Per maggiori dettagli su come fare il collegamento, dovresti consultare il Manuale della Stampante.

Prima di poter mandare una cosa qualsiasi sulla stampante, devi APRIRE un canale attraverso il quale il computer manda informazioni alla stampante. Per farlo devi usare questo comando:

```
OPEN 1,4
```

In questo modo dici al computer di aprire un canale verso il dispositivo numero 4, la stampante, e dargli il numero di file 1. Quando vuoi mandare qualcosa alla stampante devi ricollegarti ad essa tramite il numero di file che hai scelto che può essere un numero qualunque tra 1 e 255; perciò puoi usare anche il comando:

```
OPEN 54,4
```

Il motivo per cui devi usare un numero di file è che puoi avere più di un canale aperto alla volta e dando ad ogni canale che apri, un numero di file, puoi fare dei cambiamenti più facilmente, come vedrai più avanti. Può essere aggiunto un terzo numero alla fine della istruzione OPEN. È l'*indirizzo secondario* e può essere uno zero o un sette quando usi una stampante. Scegliendo un indirizzo secondario di 0 dici al computer di stampare con la maiuscola, mentre con un indirizzo secondario di 7 dici al computer di stampare con la minuscola.

Dopo aver aperto un canale, devi decidere quello che vuoi mandare alla stampante. Se vuoi stampare solo pochi messaggi devi usare il comando PRINT #. Per es.:

```
OPEN 1,4:PRINT#1,"QUESTA E' UNA PROVA DI STAMPA"
```

Appena premi il tasto RETURN, verrà stampato il messaggio QUESTA È UNA PROVA DI STAMPA. Se hai già aperto il canale con il numero di file 1, allora riceverai un messaggio d'errore. Se ciò succede digita CLOSE 1 e parti di nuovo.

Avrai osservato che abbiamo usato il numero di file che avevamo usato per aprire (OPEN) il canale nel comando PRINT #. Ciò avviene perché ci riferiamo sempre alla stampante con il suo numero di file e non con il suo dispositivo.

Puoi stampare qualunque cosa sulla stampante, come potevi fare sullo schermo, ma i comandi PRINT USING e PRINT TAB non funzionano nello stesso modo. A proposito di questi comandi ti verranno dati maggiori dettagli in seguito.

Quando hai finito di stampare i messaggi devi chiudere (CLOSE) il canale in questo modo:

```
CLOSE 1
```

Questo comando chiude il canale che ha il numero di file 1 in modo che le informazioni non possono essere più mandate alla stampante.

Il secondo modo per mandare messaggi alla stampante è di usare il comando CMD, il quale fa in modo che venga mandato alla stampante tutto ciò che

normalmente appare sullo schermo. Così se scrivi un breve programma e poi digiti:

```
OPEN 1,4:CMD 1:LIST
```

il computer aprirà un file con la stampante, dando il numero di file 1 e poi stampando il programma non sullo schermo ma sulla carta. D'ora in poi tutto ciò che il computer normalmente manda sullo schermo, verrà invece mandato alla stampante. Così se digiti:

```
PRINT "QUESTA E' UN'ALTRA PROVA DI STAMPA"
```

il messaggio QUESTA È UN'ALTRA PROVA DI STAMPA verrà stampato sulla stampante.

Quando il programma è stato listato devi chiudere di nuovo il canale, ma prima devi dire al computer di fermare l'invio di informazioni alla stampante e di mandarle di nuovo allo schermo. Perciò devi digitare:

```
PRINT#1
```

Questo comando dirà al computer di ricominciare a mandare tutte le informazioni sullo schermo, ma il canale della stampante è ancora aperto. Per chiuderlo devi usare di nuovo il comando CLOSE in questo modo:

```
CLOSE 1
```

Uso di file su nastro

È abbastanza utile poter immagazzinare dati su nastro, specialmente quando stai scrivendo un programma lungo e hai poca memoria a disposizione. Fortunatamente è possibile salvare informazioni su nastro e caricarle poi tramite variabili.

Prima di salvare qualunque informazione su nastro, devi aprire un canale su nastro, proprio come aprivi un canale verso la stampante; perciò digita:

```
10 SCNCLR
20 OPEN 3,1,2,"FILE 1"
```

Il comando OPEN sulla linea 20 dice al computer di aprire un canale verso il registratore (dispositivo numero 1) e dargli il numero di file 3. Il secondo numero corrisponde al numero del dispositivo (1 = Registratore) e il terzo dice al computer cosa vuoi fare con il file. Questo numero può essere:

0 Leggere dal nastro (INPUT);
1 Scrivere sul nastro (OUTPUT);
2 Scrivere sul nastro con segnale di fine nastro (EOT).

Nel nostro comando OPEN abbiamo scelto il 2 (Scrittura con segnale di fine nastro) per dire al computer che quando ha finito di immagazzinare informazioni sul nastro deve mettere un contrassegno sul nastro per indicare che non ci sono più informazioni in quel file.

Se questo segnale non venisse messo, il computer continuerebbe la ricerca di informazioni anche quando avesse raggiunto la fine del file. Ora, dopo aver aperto il file, dobbiamo mandare qualcosa. Usiamo il comando PRINT = per far uscire i dati verso il nastro, proprio come mandiamo i dati da scrivere verso la stampante. Aggiungi queste linee al tuo programma:

```
30 PRINT#3,"CIAO";CHR$(13);"A";CHR$(13);"TUTTI"
40 PRINT#3,1;CHR$(13);2;CHR$(13);3
50 CLOSE 3
```

La linea 30 del programma dice al computer di mandare le parole CIAO,A e TUTTI al file numero 3 che è quello aperto verso il registratore. Ogni parola è separata da un CHR$(13) che equivale a premere il tasto RETURN. Questo codice di carattere viene aggiunto per separare le parole sul nastro. Il computer ha bisogno di questo codice di carattere, che funziona da separatore, per evitare confusione di dati quando vengono caricati di nuovo dal nastro.

La linea 40 è simile alla 30 tranne per il fatto che i numeri 1, 2 e 3 sono stati usati al posto delle parole. I numeri sono ancora separati da CHR$(13).

Quando abbiamo finito di immagazzinare le informazioni sul nastro, dobbiamo chiudere il canale ed è ciò che fa la linea 50. Il comando CLOSE viene qui usato esattamente nello stesso modo in cui era stato usato per chiudere il canale verso la stampante.

Ora per eseguire il programma inserisci una cassetta vuota nel registratore, poi fai riavvolgere il nastro fino all'inizio. Ora digita RUN e premi il tasto PLAY e RECORD sul registratore. Lo schermo diventerà bianco mentre il computer immagazzina le informazioni nel programma sul nastro.

Quando le informazioni sono state registrate digita NEW e poi inserisci questo programma:

```
10 SCNCLR
20 OPEN 3,1,0,"FILE 1"
30 INPUT#3,A$,B$,C$
40 INPUT#3,A,B,C
50 CLOSE 3
60 PRINT A$;" ";B$;" ";C$
70 PRINT A,B,C
```

La linea 20 apre un canale di lettura verso il registratore. Il primo numero del comando OPEN è il numero di file, il secondo è il numero del dispositivo e il terzo dice al computer che vuoi leggere informazioni da quel dispositivo. FILE 1 è il nome del file.

Le linee 30 e 40 leggono le informazioni dal nastro usando l'istruzione INPUT #. Questa istruzione dice al computer di leggere le informazioni da un dispositivo o dalla tastiera e funziona in modo simile alla normale istruzione INPUT. In questo caso è stato detto al computer di leggere tre stringhe, che devono essere immagazzinate nelle variabili stringa A$, B$ e C$ e nella linea 40 diciamo al computer di leggere i 3 numeri, che devono essere immagazzinati nelle variabili A, B e C.

Usiamo il comando CLOSE per chiudere di nuovo il canale ed infine le linee 60 e 70 mostrano il contenuto delle variabili A$, B$ C$, A, B e C.

Ora fai riavvolgere il nastro e digita RUN. Il computer ti dirà... PRESS PLAY ON TAPE (premi il tasto PLAY sul registratore). Appena fatto lo schermo diventerà bianco, mentre il computer legge le informazioni dal nastro. Lo schermo ritornerà normale e le informazioni lette dal nastro appariranno sullo schermo.

Un altro mezzo per richiamare informazioni dal registratore è di usare il comando GET #. È simile al comando INPUT # tranne per il fatto che legge un solo carattere alla volta. Prova a digitare questo programma:

```
10 SCNCLR
20 OPEN 3,1,0,"FILE 1"
30 FOR N=1 TO 26
40 GET#3,A$
50 PRINT A$;:NEXT
60 CLOSE 3
```

Quando lo esegui, ti verrà ancora chiesto di premere il tasto PLAY sul registratore, ma questa volta il computer leggerà ciò che c'è sul nastro un carattere alla volta e lo fa apparire poco per volta (mentre va avanti).

L'istruzione GET # nella linea 40 legge informazioni dal nastro un solo carattere per volta e assegna ogni carattere alla variabile A$. Questo provesso appare sullo schermo.

Il comando GET # legge un carattere alla volta e legge anche ogni CHR$(13), perciò quando questo carattere appare sullo schermo si comporta come se avessi premuto il RETURN.

Riassunto

L'istruzione OPEN viene usata per aprire un canale per fare in modo che le informazioni possano essere mandate al dispositivo.

L'istruzione PRINT # viene usata per mandare informazioni in un canale aperto (OPENED) in precedenza.

Le informazioni possono essere lette da un dispositivo usando l'istruzione INPUT #. Essa legge subito una intera serie di caratteri.

In alternativa puoi usare la istruzione GET # che legge un solo carattere alla volta.

Quando hai finito di usare il dispositivo devi chiudere (CLOSE) il canale.

Uso di file su dischetto

Anche se è utile poter immagazzinare informazioni su nastro, è una operazione lenta. I disk drives sono molto veloci e sono ideali per l'uso dei file.

Salvare informazioni su dischetto è simile alla operazione di salvataggio di informazioni su nastro. Prova a digitare questo breve programma:

```
10 SCNCLR
20 OPEN 1,8,2,"FILE DISCO,S,W"
30 PRINT#1,"QUESTO E' UN BREVE MESSAGGIO";CHR
   $(13);"E QUESTO E' UN ALTRO"
40 PRINT#1,1;CHR$(13);2;CHR$(13);3
50 CLOSE 1
```

Il comando OPEN viene usato in modo simile sia per aprire un canale verso il disk drive sia per aprirlo verso il registratore. Il primo numero è il numero di file, il secondo è il numero del dispositivo (di solito è 8 per il disk drive) e il terzo è il canale dei data, che può essere un numero qualsiasi da 2 a 14. Le lettere tra virgolette costituiscono il nome del file, a parte ',S,W'. La 'S' dice al computer che vuoi usare un file sequenziale, in altre parole, un file dove tutte le informazioni siano immagazzinate nell'ordine in cui le avevi salvate. La 'W' dice invece che vuoi scrivere (WRITE), cioè mandare informazioni all'esterno.

Le informazioni vengono mandate al disk drive nello stesso modo in cui vengono mandate al registratore. Le linee 30 e 40 spediscono le informazioni usando l'istruzione PRINT # e ancora una volta le informazioni vengono separate da CHR$(13).

Naturalmente devi chiudere (CLOSE) il file quando hai finito con la linea 50.

Quando fai girare il programma, il disk drive partirà e le informazioni verranno registrate. Non ci vorrà molto data la velocità del drive.

Leggere le informazioni di nuovo dal dischetto è facile tanto quanto registrarle. Digita NEW e poi inserisci questo programma:

```
10 OPEN 1,8,2,"FILE DISCO,S,R"
20 INPUT#1,A$,B$,A,B,C
30 CLOSE 1
40 PRINT A$:PRINT B$:PRINT A;B;C
```

La linea 10 apre (OPEN) il canale con il drive come l'aveva fatto per il primo programma, l'unica differenza è che al nome del file seguono le lettere ',S,R', che indicano al computer che vuoi leggere (READ) da un file sequenziale (SEQUENTIAL).

Le informazioni vengono lette dalla linea 20 usando l'istruzione INPUT # nello stesso modo in cui vengono lette dal registratore. Il file viene chiuso (CLOSED) dalla linea 30 prima che la linea 40 mostri le informazioni che ha appena letto.

Come con i file su nastro, puoi usare l'istruzione GET anche per leggere le informazioni dal dischetto. Prova questo programma:

```
10 OPEN 1,8,2,"FILE DISCO,S,R"
20 FOR N=1 TO 62
30 GET#1,A$:PRINT A$;
40 NEXT:CLOSE 1
```

L'istruzione GET funziona esattamente nello stesso modo sia con i file su dischetto che con i file su nastro, come vedrai facendo eseguire questo programma.

Riassunto

Quando apri (OPEN) un file di dati con il drive devi specificare se vuoi un file sequenziale, se vuoi leggere dal dischetto o scrivere sul dischetto.

L'istruzione INPUT # viene usata per leggere delle lunghe parti di informazioni, come con i file su nastro.

L'istruzione GET # viene usata per leggere un solo carattere alla volta, sempre con file su nastro.

Il canale che stai usando deve essere chiuso (CLOSED) quando hai finito di usarlo.

APPENDICE A

ELENCO DELLE PAROLE IN BASIC COMANDI, ISTRUZIONI E FUNZIONI

Ecco un elenco di tutti i comandi, le istruzioni e le funzioni che hai studiato, insieme ad una breve descrizione di ciò che fanno. Se viene usata una lettera minuscola ad es. AB$(a), ciò indica che questa lettera può essere sostituita da un qualsiasi numero. Se una sezione di un comando è chiusa tra parentesi quadre, ad es. CIRCLE s,xr [ry,sa,ea,ra,de] ciò indica che la sezione è facoltativa.

ABS(a) Considera il valore assoluto di a, cioè lo rende positivo.

AND Può essere usato con IF ... THEN ad es. IF A = 1 AND B = 1 THEN PRINT "A" farà apparire la lettera "A" solo se il valore di A è 1 AND(e) il valore di B è 1. AND è anche un operatore Booleano, ad es. PRINT3 AND2, farà apparire il valore 2.

ASC("a") Dà il codice ASCII del carattere a.

ATN (a) Dà l'arcotangente dell'angolo a.

AUTO [n] Numerazione automatica di linea. Il valore n è l'incremento dei numeri di linea. Digitando AUTO senza alcun incremento, si cancella la numerazione automatica di linea.

BACKUP Dn TO Dm [,ON Uz] Copia il contenuto del dischetto nel drive numero n sul dischetto nel drive numero m. Puoi anche specificare di quale unità drive si tratta.

BOX fc,x1,y1,x2,y2 Disegna un rettangolo sullo schermo nella fonte di colore fc, con un angolo alle coordinate x1,y1 e l'angolo opposto alle coordinate x2,y2.

CHAR fc,x,y,"stringa" Mostra i caratteri stringa nella fonte di colore fc, con la prima lettera della stringa nella posizione di carattere x,y.

CHR$(x) Dà il carattere che ha il numero di codice x.

CIRCLE fc,x,y,xr [,yr,ap,af,ar,gr] Disegna una circonferenza o una ellissi nella fonte di colore fc, con il centro alle coordinate x,y, di raggio xr.yr è il raggio y e se non è specificato è pari a xr. Può essere disegnato un arco specificando

l'angolo di partenza ap e l'angolo finale af. La figura può essere fatta ruotare specificando l'angolo di rotazione ar e possono essere disegnate figure differenti dando il numero di gradi gr tra ciascun segmento.

CLOSE f Chiude il file f.

CLR Azzera tutte le variabili.

CMD f Fa in modo che tutte le informazioni vengano mandate al file f invece che allo schermo.

COLLECT [Dn, ON Uz] Stringe tutti i file sul dischetto e rimuove tutti i file chiusi impropriamente. Il numero del drive e l'unità drive devono essere specificate.

COLOR fc,cl,lu Assegna il colore cl al livello di luminosità lu, alla fonte di colore fc.

CONT Fa in modo che il programma riparta dopo che è stato fermato usando le istruzioni STOP o END o se è stato premuto il tasto RUN/STOP. Se è stato fatto un errore o una linea è stata modificata non puoi far proseguire il programma.

COPY [Dn,] "file 1" TO [Dm,] "file 2" [,ON Uz] Fa una copia del programma sia sullo stesso dischetto che su un altro dischetto in un altro disk drive.

COS (n) Dà il risultato del coseno dell'angolo n.

DATA elenco di dati Usato per immagazzinare un elenco di informazioni che possono venire lette di nuovo se è necessario.

DEC ("ne") Converte il numero esadecimale ne in decimale.

DEF FNva Definisce una funzione con il nome di FNva (dove 'va' è un nome qualsiasi di variabile).

DELETE linea di partenza - ultima linea Cancella blocchi di linee.

DIM elenco di variabili Riserva memoria sufficiente per le matrici (variabili numeriche dimensionate) specificate nell'elenco.

DIRECTORY [Dn,Uz,"nome file"] Mostra il contenuto del dischetto nel drive n sull'unità dischi z. Se 'il nome file' viene usato, il computer mostrerà tutti i files con questo nome.

DLOAD "nome programma" Carica un programma dal dischetto con il nome "nome programma".

DO [UNTIL argomento Booleano/WHILE argomento Booleano] linee di programma [EXIT] LOOP [UNTIL argomento Booleano/WHILE argomento Booleano] Esegue tutto ciò che è tra le istruzioni DO e LOOP fino a quando (UNTIL) o mentre (WHILE) l'argomento Booleano è diverso da zero. L'argomento Booleano può essere una condizione.

DRAW [fc[x1,y1] TO x2,y2] / [fc [,x1,y1] TO d;a] Disegna una linea dal punto x,y al punto x2,y2 nella fonte di colore fc. Questo comando può essere usato in alternativa per disegnare (DRAW) una linea lunga d pixel, inclinata di a gradi.

DSAVE "nome programma" Salva il programma in memoria sul dischetto sotto il nome di "nome programma".

ELSE istruzioni Usata con la struttura IF ... THEN. Vedi IF.

END Dice al computer di fermarsi nell'esecuzione del programma.

ERR$ (ne) Dà il messaggio d'errore che corrisponde al numero di errore ne.

EXIT Usato per uscire da un ciclo DO ... LOOP.

EXP (x) Dà il valore di e elevato alla potenza di x.

FRE (x) Dà la quantità di memoria disponibile per programmi BASIC in byte. Il valore x può essere un numero qualsiasi.

FORn=pa TO fi [STEP in]: istruzioni:NEXTn Esegue tutte le istruzioni tra le istruzioni FOR e NEXT fino a quando il valore della variabile raggiunge il valore fi.n parte dal valore pa ed è aumentato di 1 (o del valore di in) fino a quando raggiunge il valore di fi.

GET [KEY] var$, var$,var$... Controlla la tastiera e immagazzina i caratteri di qualunque tasto sia stato premuto nella variabile stringa var$. Se viene aggiunta l'istruzione KEY (ad es. GETKEY var$) allora il computer fermerà l'esecuzione del programma fino a quando il tasto è premuto. Sia l'istruzione GET che GETKEY possono essere usate per controllare parecchi tasti premuti aggiungendo un elenco di variabili dopo l'istruzione.

GRAPHIC modo [,pul]/CLR È usato per selezionare un modo grafico. Il valore deve essere tra 0 e 4 e 'pul' può essere sia 0 (per lasciare lo schermo così com'è) o 1 (per pulirlo). L'istruzione GRAPHIC CLR dà alla memoria lo spazio che precedentemente era stato riservato per i grafici, perché lo possa usare per i programmi normali.

GSHAPE var$ [,x,y,z] Mostra sullo schermo la forma grafica immagazzinata nella variabile stringa var$. Se non è stata specificata nessuna posizione, la figura appare con l'angolo in alto a sinistra posizionato sul cursore pixel. Le coordinate dell'angolo in alto a sinistra (x e y) devono essere specificate. Il valore z è facoltativo ed indica il modo di visualizzazione.

GOTO n Fa in modo che il computer salti alla linea numero n in un'altra parte del programma per continuare ad eseguire il programma da quel punto.

GOSUB n Simile al GOTO eccettuato il fatto che manda il computer ad una sottoprocedura che inizia alla linea n. Quando raggiunge una istruzione RETURN il computer ritorna all'istruzione immediatamente dopo il GOSUB.

HEADER "nome" ,Iid,Drive,ON Uunit Questo comando formatta un dischetto; questa operazione deve essere fatta prima che il dischetto venga usato. Se viene omesso il numero di identificazione, viene eseguita una formattazione veloce e viene pulita solo la directory. Può essere fatto solo con un dischetto già formattato.

HELP Se avviene un errore in un programma BASIC, questo comando mostrerà la linea sbagliata con l'errore lampeggiante sullo schermo.

HEX$ (n) Converte un numero decimale n in esadecimale e restituisce una stringa.

IF argomento Booleano THEN istruzioni [:ELSE istruzioni] L'istruzione che segue il THEN verrà eseguita se la condizione definita dall'argomento Booleano è verificata. Se non è così verrà eseguita la condizione che segue all'istruzione ELSE.

INPUT ["stringa";] var1, var2, var3... Mostra la stringa 'stringa' (se viene aggiunta) e poi aspetta che venga digitato qualcosa, che verrà poi assegnato alla variabile var (può essere una stringa, una variabile numerica intera o dimensionata). Molte informazioni possono essere inserite mettendo le variabili dopo l'istruzione INPUT.

INPUT# f, var,var,var... Simile a INPUT tranne perché legge le informazioni dal file f.

INSTR (stringa1,stringa2 [,in] Fa ritornare la posizione della stringa 'stringa2' nella stringa 'stringa1'. Se viene dato il valore di partenza in, la ricerca partirà da quella posizione.

INT (x) Arrotonda per difetto il numero x.

JOY (x) Dà un valore a seconda della posizione del joystick x. Se è premuto il pulsante del fuoco, viene aggiunto 128 al valore direzionale.

KEY [n,"comando"] Mostra i comandi che sono stati assegnati a ciascun tasto funzione, che può anche essere ridefinito aggiungendo n e il 'comando' da assegnargli.

LEFT$ (stringa,n) Dà gli ultimi n caratteri della stringa 'stringa'.

LEN (stringa) Dà il numero dei caratteri nella stringa 'stringa'.

LET var = valore Può essere usato per assegnare un valore alla variabile, ma non è necessario.

LIST [linea di partenza - linea di arrivo] Mostra tutto o parte del programma sullo schermo.

LOAD "nome del programma" [,dispositivo,ri] Carica il programma con il nome "nome programma" nella memoria del registratore se non è stato specificato

altrimenti. Se il valore di ri è 1 il programma viene caricato nella zona di memoria da cui era stato salvato. Viene usato normalmente per programmi in L.M.

LOCATE x,y/d;a Sposta il cursore pixel alle coordinate x,y. In alternativa il cursore pixel può essere mosso di d pixel con un angolo di a gradi.

LOG (x) Dà il logaritmo naturale del valore x.

LOOP Determina la fine del ciclo DO ... LOOP. Vedi DO.

MID$ (stringa$,x,n) Dà n caratteri dal centro della stringa partendo con il carattere alla posizione x.

MONITOR Inserisce TEDMON

NEW Cancella il programma e le variabili attualmente in memoria.

NEXT [var] Segna la fine del ciclo FOR ... NEXT. Vedi FOR.

NOT n Questo è un operatore Booleano e dà il valore contrario a n. Può anche essere usato con IF ... THEN (per es. IF NOT A = 1 THEN PRINT "SI" solo se A non è uguale a 1).

ON n GOTO/GOSUB linea1, linea 2, linea3... A seconda del valore della variabile n il computer salterà ad una delle linee del programma nell'elenco.

OPEN f, per [,indirizzo secondario,"nome file,tipo file,modo"] Apre un canale con un dispositivo (periferica) dandogli il numero di file f e nome 'nome file'.

OR Usato con IF ... THEN (ad es. IF A=1 OR B=1 THEN STOP fermerà il programma sia che il valore di A sia 1 o B sia 1 o entrambi siano uguali a 1).

PAINT [fc,x,y,modo] Riempie un'area dello schermo con un colore. Se non vengono specificate le coordinate di partenza, il colore parte dalla posizione del cursore pixel. Se non è specificata la fonte di colore fc, viene usato il colore dello sfondo.

PEEK (ind) Dà il contenuto della locazione di memoria ind.

POKE ind, val Immagazzina val nella locazione ind.

POS (x) Dà la posizione orizzontale del cursore; x può essere un valore qualsiasi.

PRINT lista Mostra il contenuto dell'elenco sullo schermo (ad es. PRINT "CIAO" – mostra la parola 'CIAO' sullo schermo, PRINT 4+5 mostra il risultato dell'operazione 4+5).

PRINT# f, lista Simile al PRINT tranne perché questa istruzione manda il contenuto della lista al file f e non allo schermo.

PUDEF "caratteri" Viene usato per ridefinire i simboli usati con l'istruzione PRINT USING.

RCLR (n) Dà il colore assegnato alla fonte di colore n.

RDOT (n) Dà le informazioni sul cursore pixel (n=0 per la posizione x del cursore pixel, n=1 per la posizione y del cursore pixel, n=2 per la fonte di colore).

READ var,var,var Mette le informazioni dell'istruzione DATA nella variabile che segue l'istruzione READ. var può essere una variabile qualsiasi.

RENAME "vecchio" TO "nuovo" [,Ddrive,Uunit] Cambia il nome del file sul dischetto dal vecchio nome al nuovo nome.

RENUMBER[ni,in,vi] Rinumera un programma partendo dalla vecchia linea vi, che diventerà ni e prosegue con incremento in. Se ni, in e vi non sono specificati, partirà dalla prima linea che diventerà la 10 con incremento di 10 in 10.

REM messaggio È usato per aggiungere dei commenti al programma. Il messaggio dopo l'istruzione REM viene ignorato.

RESTORE [n] Dice al computer di incominciare a leggere dati dalla prima istruzione DATA. Se viene specificato n il computer inizia a leggere dati dall'istruzione DATA sulla linea n.

RESUME [n/NEXT] Dopo che viene eseguita una istruzione TRAP, questa istruzione farà in modo che il computer tenti di eseguire una seconda volta l'istruzione che ha causato l'errore. RESUME NEXT farà in modo che il computer esegua l'istruzione che segue quella che ha causato l'errore. Se viene specificato n, il computer partirà con il programma dalla linea n.

RETURN Questa istruzione dice al computer di tornare all'istruzione dopo l'ultimo GOSUB che è stato eseguito.

RGR (n) Dà il modo grafico corrente. n è l'argomento fittizio e non ha alcun valore.

RIGHT$ (stringa,n) Dà i primi n caratteri della stringa stringa.

RLUM (fc) Dà il livello di luminosità della fonte di colore fc.

RND (argomento) Dà un valore tra 0 e 1. Il modo in cui viene scelto il numero, è definito dall'argomento.

RUN [n] Fa partire il programma in memoria. Se viene specificato n, il programma viene fatto partire dalla linea n.

SAVE "nome programma" [,dispositivo, EOT] Salva il programma corrente in memoria sul nastro, se non viene specificato un altro numero di dispositivo. Se viene aggiunto ,1 alla fine, viene aggiunto un segnale di fine nastro (EOT).

SCALE1/0 Inserisce o disinserisce la facilitazione SCALE.

SCNLR Pulisce lo schermo.

SCRATCH "nome file" [,Ddrive,Uunit] Cancella un programma da un dischetto.

SGN (n) Dà il valore 1,–1 o 0 a seconda che n sia positivo, negativo o nullo.

SIN (a) Dà il seno dell'angolo a, che deve essere in radianti.

SOUND voce, nota, durata Suona la nota n con la voce v per una durata d.

SPC (n) Simile a TAB eccettuato il fatto che funziona anche sulla stampante, mentre TAB no.

SQR (n) Dà la radice quadrata di n.

SSHAPE var$,x1,y1 [x2,y2] Immagazzina in var un'area dello schermo dalle coordinate x1,y1 alle coordinate x2,y2. Se x2 e y2 non vengono specificate, vengono prese alla posizione del cursore pixel.

STEP In Usato con l'istruzione FOR. Vedi FOR.

STOP Ferma l'esecuzione di un programma, con un messaggio BREAK (interruzione).

STR$ (n) Converte il valore n in una stringa.

STS indirizzo Esegue il programma in L.M. partendo dall'indirizzo di locazione della memoria.

TAB (n) Muove il cursore alla casella n + 1 in orizzontale.

TAN (a) Dà la tangente del'angolo a che deve essere in radianti.

THEN istruzioni Parte della struttura IF ... THEN ... ELSE. Vedi IF.

TO Usato con l'istruzione FOR. Vedi FOR.

TRAP n Fa saltare il computer alla linea n quando si imbatte in un errore.

TROFF Disinserisce la facilitazione TRACCIA.

TRON Inserisce la facilitazione TRACCIA.

UNTIL argomento Booleano Usato nel ciclo DO ... LOOP. Vedi DO.

USING stringa; elenco Usato con il PRINT per mostrare i contenuti dell'elenco in un formato specificato dalla stringa.

var=USR (x) Inizia una routine in L.M. il cui indirizzo di partenza è immagazzinato nelle locazioni di memoria 1281 e 1282. Il valore x viene passato al programma in L.M. nell'accomulatore a virgola fluttuante. La variabile var contiene un numero che viene passato dal L.M.

VAL ("stringa") Dà il valore numerico della stringa 'stringa'.

VERIFY "nome programma" [,periferica, indicatore di rilocamento] Confronta il programma sul nastro (se non viene specificata una periferica differente) con il programma in memoria. Se i programmi sono uguali, appare il messaggio OK, altrimenti appare il messaggio VERIFY ERROR (errore nella verifica).

VOL v Inserisce il livello del volume al valore v.

WAIT indirizzo, valore1, valore2 L'esecuzione del programma viene fermata fino a quando il contenuto della locazione in AND il valore 1 (EXOR valore 2) è diverso da zero.

WHILE argomento Booleano Viene usato nel ciclo DO ... LOOP. Vedi DO.

APPENDICE B

ABBREVIAZIONI IN BASIC

Molti dei comandi delle istruzioni e delle funzioni che il tuo computer comprende, hanno delle forme abbreviate. Di solito sono formate dai primi due o tre caratteri dell'istruzione seguiti da un carattere + SHIFT. Qui di seguito ecco un elenco delle abbreviazioni. Il carattere chiuso tra parentesi quadre deve essere scritto tenendo premuto lo SHIFT.

ABS	A[B]	A│	GET	G[E]	G‾
ASC	A[S]	A♥	GETKEY	GETK[E]	GETK‾
ATN	A[T]	A│	GET#	NONE	GET#
AUTO	A[U]	A╱	GOSUB	GO[S]	GO♥
BACKUP	B[A]	B♠	GOTO	G[O]	G┌
BOX	B[O]	B┌	GRAPHIC	G[R]	G_
CHAR	CH[A]	CH♠	GSHAPE	G[S]	G♥
CHR$	C[H]	C│	HEADER	HE[A]	HE♠
CIRCLE	C[I]	C╲	HELP	HE[L]	HEL
CLOSE	CL[O]	CL┌	HEX$	H[E]	H‾
CLR	C[L]	CL	IF	NONE	IF
CMD	C[M]	C╲	INPUT	NONE	INPUT
COLLECT	COL[L]	COLL	INPUT#	I[N]	I╱
COLOR	CO[L]	COL	INSTR	IN[S]	I╱
CONT	C[O]	C┌	INT	NONE	INT
COPY	CO[P]	CO┐	JOY	J[O]	J┌
COS	NONE	COS	KEY	K[E]	K‾
DATA	D[A]	D♠	LEFT$	LE[F]	LE_
DEC	NONE	DEC	LEN	NONE	LEN
DEF	D[E]	D‾	LET	L[E]	L‾
DELETE	DE[L]	DEL	LIST	L[I]	L╲
DIM	D[I]	D╲	LOAD	L[O]	L┌
DIRECTORY	DI[R]	DI_	LOCATE	LO[C]	LO‾
DLOAD	D[L]	DL	LOG	NONE	LOG
DO	NONE	DO	LOOP	LO[O]	LO┌
DRAW	D[R]	D_	MID$	M[I]	M╲
DSAVE	D[S]	D♥	MONITOR	M[O]	M┌
ELSE	E[L]	EL	NEW	NONE	NEW
END	E[N]	E╱	NEXT	N[E]	N‾
ERR$	E[R]	E_	NOT	N[O]	N┌
EXIT	EX[╲]	EX╲	ON	NONE	ON
EXP	E[X]	E♣	OPEN	O[P]	O┐
FOR	F[O]	F┌	PAINT	P[A]	P♠
FRE	F[R]	F_	PEEK	P[E]	P‾

POKE	P[O]	P┌	SOUND	S[O]	S┌
POS	NONE	POS	SPC	S[P]	S┐
PRINT	?	?	SQR	S[Q]	S●
PRINT#	P[R]	P_	SSHAPE	S[S]	S♥
PUDEF	P[U]	P╭	ST	NONE	ST
RCLR	R[C]	R−	STEP	ST[E]	ST‾
RDOT	R[D]	R‾	STOP	S[T]	S|
READ	R[E]	R‾	STR$	ST[R]	ST_
REM	NONE	REM	SYS	S[Y]	S|
RENAME	RE[N]	RE╱	TAB	T[A]	T♠
RENUMBER	REN[U]	REN╭	TAN	NONE	TAN
RESTORE	RE[S]	RE♥	THEN	T[H]	T|
RESUME	RES[U]	RES╭	TI	NONE	TI
RETURN	RE[T]	RE|	TI$	NONE	TI$
RGR	R[G]	R|	TRAP	T[R]	T_
RIGHT$	R[I]	R╮	TROFF	TRO[F]	TRO_
RLUM	R[L]	RL	TRON	TR[O]	TR┌
RND	R[N]	R╱	UNTIL	U[N]	U╱
RUN	R[U]	R╭	USING	US[I]	US╮
SAVE	S[A]	S♠	USR	U[S]	U♥
SCALE	SC[A]	SC♠	VAL	NONE	VAL
SCNCLR	S[C]	S−	VERIFY	V[E]	V‾
SCRATCH	SC[R]	SC_	VOL	V[O]	V┌
SGN	S[G]	S|	WAIT	W[A]	W♠
SIN	S[I]	S╮	WHILE	W[H]	W|

APPENDICE C

CODICE ASCII

Qui di seguito un elenco di codici ASCII per tutti i caratteri disponibili sul tuo computer. Come avrai probabilmente notato, alcuni codici non hanno alcun effetto e altri compaiono 2 volte.

0		32	SPAZIO	64	@	96	─
1		33	!	65	A	97	♠
2		34	"	66	B	98	│
3		35	#	67	C	99	─
4		36	$	68	D	100	─
5	BIANCO	37	%	69	E	101	▔
6		38	&	70	F	102	_
7		39	'	71	G	103	│
8	DISAB. SH+CBM	40	(	72	H	104	│
9	ABIL. SH+CBM	41	)	73	I	105	╮
10		42	*	74	J	106	╰
11		43	+	75	K	107	╯
12		44	,	76	L	108	L
13	RETURN	45	-	77	M	109	╲
14	MINUSCOLO	46	.	78	N	110	╱
15		47	/	79	O	111	┌
16		48	0	80	P	112	┐
17	CURSORE GIU'	49	1	81	Q	113	●
18	REVERSE ON	50	2	82	R	114	_
19	HOME	51	3	83	S	115	♥
20	DELETE	52	4	84	T	116	│
21		53	5	85	U	117	╭
22		54	6	86	V	118	╳
23		55	7	87	W	119	○
24		56	8	88	X	120	♣
25		57	9	89	Y	121	│
26		58	:	90	Z	122	♦
27		59	;	91	[	123	┼
28	ROSSO	60	<	92	£	124	▒
29	CURS. DESTRA	61	=	93	]	125	│
30	VERDE	62	>	94	↑	126	π
31	BLU	63	?	95	←	127	◥

128		160	SH+SPAZIO	192	─	224	SH+SPAZIO
129	ARANCIO	161	▌	193	♠	225	▌
130	FLASH ON	162	▄	194	│	226	▄
131		163	▔	195	─	227	▔
132	FLASH OFF	164	_	196	─	228	_
133		165	|	197	─	229	|
134		166	▒	198	─	230	▒
135		167	|	199	│	231	|
136		168	▒	200	│	232	▒
137		169	◤	201	╮	233	◤
138		170	▐	202	╰	234	▐
139		171	├	203	╯	235	├
140		172	▗	204	└	236	▗
141	SH+RETURN	173	└	205	╲	237	└
142	MAIUSCOLO	174	┐	206	╱	238	┐
143		175	▂	207	┌	239	▂
144	NERO	176	┌	208	┐	240	┌
145	CURSORE SU	177	┴	209	●	241	┴
146	REVERSE OFF	178	┬	210	_	242	┬
147	CLEAR SCREEN	179	┤	211	♥	243	┤
148	INSERT	180	▎	212	|	244	▎
149	MARRONE	181	▍	213	╭	245	▍
150	GIALLO-VERDE	182	▐	214	╳	246	▐
151	ROSA	183	▔	215	○	247	▔
152	BLU-VERDE	184	▀	216	♣	248	▀
153	BLU CHIARO	185	▃	217	|	249	▃
154	BLU SCURO	186	┘	218	♦	250	┘
155	VERDE CHIARO	187	▖	219	┼	251	▖
156	PORPORA	188	▝	220	▒	252	▝
157	CURS. SINIS.	189	┘	221	│	253	┘
158	GIALLO	190	▘	222	π	254	▘
159	CIANO	191	▚	223	◥	255	π

READY.

APPENDICE D

CODICE DELLO SCHERMO

Ecco un elenco dei primi 128 caratteri e dei loro codici dello schermo. Gli ultimi 128 sono esattamente uguali ai primi, tranne che sono in negativo. Così per es. se vuoi la lettera H in negativo dovrai aggiungere 128 al codice della lettera H (che è 8), avendo come risultato 136. Questo è dunque il codice della lettera H in negativo.

0	@	20	T	40	(	60	<	80	┐	100	_	120	‾
1	A	21	U	41	)	61	=	81	●	101	▏	121	▬
2	B	22	V	42	*	62	>	82	–	102	▒	122	┘
3	C	23	W	43	+	63	?	83	♥	103	▕	123	▖
4	D	24	X	44	,	64	–	84	│	104	▒	124	▝
5	E	25	Y	45	-	65	♠	85	╭	105	◤	125	┘
6	F	26	Z	46	.	66	│	86	╳	106	▕	126	▘
7	G	27	[	47	/	67	–	87	○	107	├	127	▚
8	H	28	£	48	0	68	–	88	♣	108	▗		
9	I	29	]	49	1	69	‾	89	│	109	└		
10	J	30	↑	50	2	70	–	90	♦	110	┐		
11	K	31	←	51	3	71	│	91	+	111			
12	L	32		52	4	72	│	92	▒	112	┌		
13	M	33	!	53	5	73	╮	93	│	113	┴		
14	N	34	"	54	6	74	╰	94	π	114	┬		
15	O	35	#	55	7	75	╯	95	◥	115	┤		
16	P	36	$	56	8	76	L	96		116	▏		
17	Q	37	%	57	9	77	╲	97	▌	117	▍		
18	R	38	&	58	:	78	╱	98	▄	118	▐		
19	S	39	'	59	;	79	┌	99	‾	119	‾		

APPENDICE E

GLOSSARIO

Leggendo libri o riviste sui computer ti capiterà di incontrare parole di cui non sai il significato. Eccoti una breve guida per aiutarti a capire il gergo del computer.

Accoppiatore acustico È un dispositivo che può essere collegato al computer, in cui può essere inserito un apparecchio telefonico. Il tuo computer potrà perciò comunicare con un altro attraverso il telefono.

Alta risoluzione La misura della risoluzione corrisponde al numero di pixel sullo schermo. Più pixels sono possibili, più alta è la risoluzione.

BASIC (Beginners's All-Purpose Symbolic Instruction Code = Codice simbolico di istruzioni multiuso per il principiante). Questo è il linguaggio che viene compreso dalla maggior parte dei microcomputers, incluso il tuo.

Bit È la più piccola unità di informazione che può essere utilizzata da un computer e può essere 1 o 0.

Bug (BACO, errore) Un errore in un programma che non lo fa funzionare o lo fa funzionare in modo scorretto.

Byte È un numero binario che è costituito da 8 bits. Dato che ci sono 2 ↑ 8 (=256) combinazioni di 8 uno/zero, un byte può contenere un numero qualsiasi tra 0 e 255.

Cartridge Una cartuccia contenente un programma o memoria RAM aggiuntiva che può essere inserita nel computer.

Codice macchina o linguaggio macchina (L.M.) Il linguaggio che la CPU può capire. È fatto solo di numeri!

Comando Una istruzione che normalmente viene digitata direttamente senza usare i numeri di linea (ad es. RUN,NEW,LIST ecc.).

Compilatore Converte un programma scritto in linguaggio ad alto livello, quale ad es. il BASIC, in linguaggio macchina.

CP/M Control Program for Microcomputers (Programma di controllo per microcomputers). È un sistema operativo standard per dischi che è disponibile per molti dei computers basati su Z80. È un linguaggio standard che si incontra spesso nei computer aziendali. Dato che è standard, i programmi scritti in CP/M possono essere facilmente trasferiti da un computer ad un altro.

CPU Central Processing Unit (Unità centrale di elaborazione). È il cervello del computer.

Cursore È un simbolo che indica dove apparirà sullo schermo il carattere seguente.

Data Un'altra parola usata per indicare dati.

Debug Eliminazione di tutti gli errori dal programma.

Disk È un nastro magnetico circolare che è racchiuso in un involucro di protezione. I programmi e altre informazioni possono essere immagazzinate sui dischetti e rilette ad alta velocità. Sui dischetti può essere immagazzinata una grande quantità di informazioni.

DOS Disk Operating System (Sistema operativo del dischetto). È un programma contenuto sia nel computer che nel disk drive (qualche volta può essere caricato nel computer) che controlla il funzionamento dei drives.

EEPROM Electrically Erasable Programmable Read Only Memory. È uno speciale tipo di ROM che può essere programmata (usando un programmatore speciale). Può essere cancellata con impulsi elettrici.

EPROM Erasable Programmable Read Only Memory. È simile alla EEPROM, ma può essere cancellata con l'esposizione diretta ai raggi ultra-violetti.

Esadecimale (hex) È la numerazione in base 16. Le lettere dalla A alla F vengono usate in aggiunta alle cifre dallo 0 al 9. (A=10, B=11, C=12 ecc.).

Floppy disk vedi disk (dischetto).

Funzione Una istruzione che prende un numero e lo usa per eseguire dei calcoli, prima di dare il risultato.

Hard copy Stampa di informazioni prodotte da una stampante su carta.

Hard disk È simile al dischetto normale ma è fissato permanentemente al disk drive. Può immagazzinare molte più informazioni di un dischetto normale, ma è molto più dispendioso a causa della accuratezza con cui la testina di lettura viene mantenuta nella posizione corretta sul dischetto.

Hardware È il computer — la parte meccanica che puoi toccare. Vedi software.

Indirizzo È un numero che serve da indice per le locazioni di memoria.

Interfaccia È un dispositivo che permette di collegare una periferica al computer.

I/O Sta per Input/Output. Può esser usato con riferimento ad una periferica che legge le informazioni e le manda all'esterno, oppure ad una porta che può mandare e ricevere informazioni.

Istruzione Una parola che dice al computer di eseguire una operazione.

Kilobyte (K) 1024 bytes di memoria.

Linguaggio Assembly Un linguaggio di programmazione nel quale vengono usate delle istruzioni simboliche per eseguire dei processi modificando il contenuto della memoria del computer.

Mappa di memoria È una tabella che mostra quale area della memoria del computer viene usata.

Modem È un altro dispositivo che permette la comunicazione tra i computers attraverso una linea telefonica. Questo dispositivo si collega direttamente alla linea telefonica e deve essere approvato dalle Telecomunicazioni o dalla Compagnia Telefonica addetta.

Modulatore È un dispositivo normalmente collegato all'interno del computer, che converte il segnale di uscita del computer in modo da poter apparire sullo schermo della televisione.

Monitor Può essere riferito sia al dispositivo che viene usato come la televisione, ma che possiede una migliore qualità visiva, sia al programma che permette di esaminare e modificare le locazioni di memoria.

Parallelo È il modo in cui il computer legge o scrive informazioni. Una interfaccia parallela permette di usare parecchi bit di informazioni in una sola volta.

Pascal Un linguaggio efficace ad alto livello, usato su alcuni computers aziendali.

Periferica È un dispositivo che può essere collegato al computer per aumentarne le possibilità.

Pixel Il punto più piccolo che il computer può accendere sullo schermo della televisione.

Porta È un collegamento attraverso il quale le informazioni possono essere lette o scritte.

Printout Vedi Hard copy.

Programma È un gruppo di istruzioni combinate in modo da ottenere un risultato utile.

PROM Programmable Read Only Memory. Un tipo speciale di ROM che può essere programmata da un dispositivo apposito.

QWERTY Il nome dato alla disposizione standard della tastiera.

RAM Random Access Memory (Memoria ad accesso casuale). È un tipo speciale di memoria il cui contenuto non è permanente e può essere modificata. Questo tipo di memoria trattiene i suoi contenuti solo se l'alimentazione è inserita.

Registro Una parte della CPU nella quale può essere immagazzinato un numero. È simile ad una variabile.

ROM Read Only Memory. È un tipo di memoria permanente e i suoi contenuti non possono essere modificati. Non è necessario che l'alimentazione sia inserita per trattenerne i contenuti.

Routine Una parte di programma che serve per realizzare una specifica operazione.

RS232 Una forma standard di interfaccia che utilizza informazioni in forma seriale.

Seriale È un tipo di trasmissione di dati che vengono spediti un bit alla volta.

Set di caratteri (Character Set) Un gruppo di numeri, lettere e simboli che il computer può realizzare su uno schermo normale.

Software È il programma e viene immagazzinato in un dispositivo hardware.

Source code È un programma scritto con un linguaggio ad alto livello come il BASIC o il Pascal, che necessita di essere compilato in L.M.

Statement Una istruzione contenuta in un programma.

Stringa Una serie di caratteri.

Subroutine Vedi Routine.

Toolkit È un programma che aggiunge dei comandi a quelli già disponibili.

Utility Simile al Toolkit.

Variabile Un valore che può essere modificato. Viene rappresentato di solito da un carattere o una serie di caratteri.

VDU Visual Display Unit. Può essere sia una televisione che un monitor.

Z80 È un modello molto conosciuto di CPU che è il cuore del computer quali ad es. ZX Spectrum, Sharp e i computer MSX.

6502 È un altro tipo di CPU. Su di essa si basa la CPU del tuo computer.

APPENDICE F

ALCUNI PROGRAMMI IN BASIC

Per finire completamente questo libro, ecco qui di seguito alcuni brevi programmi da provare.

3-D Plot

```
 10 GRAPHIC 3,1:COLOR 3,2,7
 20 A=80:B=A*A:C=100:D=100
 30 FOR X=0 TO A
 40 S=X*X
 50 P=SQR(B-S)
 60 I=-P
 70 R=SQR(S+I*I)/A
 80 Q=(R-1)*SIN(24*R)
 90 Y=I/3+Q*D
100 IF I=-P THEN M=Y:GOTO 130
110 IF Y>M THEN M=Y:GOTO 140
120 IF Y>=N THEN GOTO 170
130 N=Y
140 Y=Y+C
150 DRAW 3,A+X,Y
160 DRAW 3,A-X,Y
170 I=I+4
180 IF I<P THEN 70
190 NEXT X
200 FOR C=2 TO 15:FOR I=0 TO 7:COLOR 3,C,I
210 FOR M=0 TO 500:NEXT M,I,C
220 GOTO 200
```

Sveglia

Quando esegui il programma “Sveglia” ti verrà chiesto a che ora vuoi la sveglia. Il programma lavora come un orologio di 24 ore, perciò l’allarme deve essere

inserito nella forma OOMMSS, dove OO è l'ora, MM sono i minuti e SS i secondi. Per inserire l'ora basta premere il tasto O per muovere la lancetta delle ore, il tasto M per muovere la lancetta dei minuti e il tasto S per muovere la lancetta dei secodi.

```
 10 SCNCLR:PRINT"A CHE ORA VUOI CHE LA SVEGLIA
     SUONI (OOMMSS)";
 20 INPUT AL$:IF LEN(AL$)<>6 THEN RUN
 30 VOL 5:V=5
 40 Z=350
 50 GRAPHIC 1,1
 60 COLOR 0,3,3:COLOR 4,3,3:COLOR 1,8,6
 70 SCNCLR
 80 CIRCLE 1,160,100,75
 90 FOR N=0 TO 360 STEP 6
100 DRAW 0,160,100 TO 72;N
110 DRAW 1 TO 3;N
120 NEXT
130 FOR N=0 TO 360 STEP 30
140 DRAW 0,160,100 TO 65;N
150 DRAW 1 TO 10;N
160 NEXT
170 X=344:GOSUB 430
180 Z=344:GOSUB 370
190 TI$="000000"
200 IF Z>704 THEN Z=344:GOSUB 370:GOSUB 430
210 FOR A=0 TO 354 STEP 6
220 SOUND 2,1000,1
230 COLOR 1,2,7
240 DRAW 1,160,100 TO 56;A
250 GOSUB 400:GOSUB 450
260 GET A$
270 IF A$="C" THEN GOSUB 470
280 IF A$="A" THEN AL=0
290 IF TI$=AL$ THEN AL=1
300 IF AL=1 THEN FOR M=1 TO 9:SOUND 1,700,2:S
    OUND 1,800,2:NEXT
310 IF AL=1 THEN 330
320 FOR M=1 TO 310:NEXT M
330 DRAW 0,160,100 TO 56;A
340 NEXT A
350 GOSUB 370
360 GOTO 200
370 DRAW 0,160,100 TO 40;Z TO 18;Z+34 TO 18;Z
    -194 TO 160,100
380 Z=Z+6
390 E=E+1
400 DRAW 1,160,100 TO 40;Z TO 18;Z+34 TO 18;Z
    -194 TO 160,100
410 IF E=12 THEN E=0:GOSUB 430
420 RETURN
430 DRAW 0,160,100 TO 30;X TO 14;X+33 TO 14;X
    -192 TO 160,100
```

```
440 X=X+6
450 DRAW 1,160,100 TO 30;X TO 14;X+33 TO 14;X
    -192 TO 160,100
460 RETURN
470 GETKEY A$
480 IF A$="O" THEN GOSUB 430:Z$=STR$(VAL(TI$)
    +10000):Z$=RIGHT$(Z$,LEN(Z$)-1)
490 IF A$="M" THEN GOSUB 370:Z$=STR$(VAL(TI$)
    +100):Z$=RIGHT$(Z$,LEN(Z$)-1)
500 IF A$="S" THEN DRAW 0,160,100 TO 56;A:Z$=
    STR$(VAL(TI$)+1)
510 DRAW 0,160,100 TO 57;A
520 IF A$="S" THEN Z$=RIGHT$(Z$,LEN(Z$)-1):A=
    A+6
530 DRAW 1,160,100 TO 57;A:GOSUB 450:GOSUB 40
    0
540 IF Z$="" THEN 570
550 IF LEN(Z$)<6 THEN Z$=RIGHT$("000000",6-LE
    N(Z$))+Z$:TI$=Z$:Z$=""
560 IF A$="T" AND V=0 THEN VOL 5:V=5:ELSE IF
    A$="T" THEN V=0:VOL 0
570 IF A$="C" THEN DRAW 0,160,100 TO 57;A:RET
    URN
580 GOTO 470
```

Poligoni

```
10 GRAPHIC 2,1
20 INPUT"QUANTI LATI";A
30 IF A<2 OR A>100 THEN PRINT"NON ESSERE RIDI
   COLO !":GOTO 20
40 SCNCLR
50 CIRCLE 1,160,80,40,33,,,,360/A
60 GOTO20
```

INDICE ANALITICO



Il libro è stato scritto appositamente per chi è completamente a digiuno di computer, e questo scopo è rimasto sempre nella mente dell'autore durante lo svolgimento dell'opera. Nonostante ciò, il libro si discosta da un usuale manuale di Basic o da un semplice studio didattico; riesce, infatti, attraverso esempi chiari e divertenti, a stimolare nel lettore il desiderio di apprendere e concretizzare l'utilizzo del computer, che presto diventerà un inseparabile amico.

L. 16.000

COD. 423B ISBN 88-7056-258-1

203

i tuoi amici commodore 16 e PLUS 4

Brian Lloyd



GRUPPO EDITORIALE JACKSON